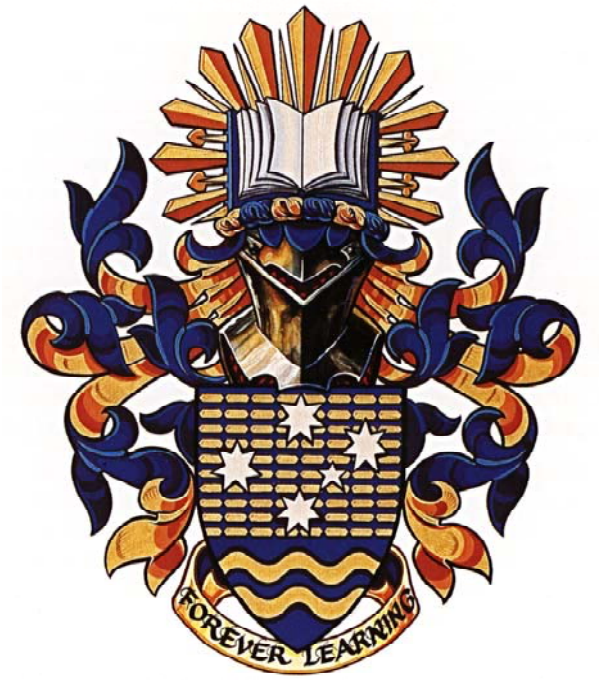
DOCTORAL THESIS

A Graphical User Interface Reference Model for Messaging Systems with Directory Integration

Iannella, Renato

Award date:
1994

Bond University



A Graphical User Interface Reference Model for Messaging Systems with Directory Integration

by

Renato Iannella

May 1994

Submitted to Bond University in fulfilment of the requirements
for the Degree of Doctor of Philosophy in Computer Science

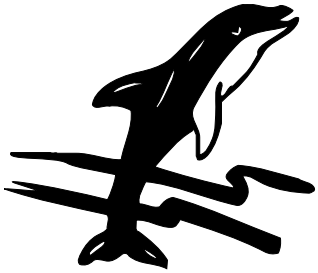
Bond University
School of Information Technology

This thesis is submitted to Bond University in fulfilment of the requirements for the Degree of Doctor of Philosophy.

This thesis represents my own work and contains no material which has been previously submitted for a degree or diploma at the University or any other institution, except where due acknowledgment is made.

Signature: _____

Date: _____



School of Information Technology

School of Information Technology
Bond University
Queensland, 4229, AUSTRALIA

ISBN 0 7331 0002 3

Dedication

This thesis is dedicated to my parents.

I am forever beholden for their love and devotion.

Acknowledgements

Numerous people need to be thanked for their support over the many years of work involved in the development of my PhD project. Firstly, and most obviously, my supervisor, Professor Michael J Rees for his guidance and encouragement throughout the entire project. His approach and feedback to my work was most gratifying.

Many thanks to Mr George Michaelson, from the Prentice Centre at the University of Queensland, for his helpful cooperation in installing the base X.400 and X.500 software used in this project.

I am also grateful to Professor David J Scott for his assistance in the statistical analysis of the usability evaluation data and Professor Graham McMahon for his review of the final thesis.

Also, I would like to thank the external examiners for suggesting revisions included in the final version of this thesis:

- Professor Gary Perlman, Ohio State University, USA
- Dr Gitte Lindgaard, Telecom Research Labs, Melbourne
- Professor Jim Welsh, University of Queensland, Brisbane

On a personal level, I would like to thank my family for their support and the understanding they have shown over the years and long distances. Also, to Ms Elizabeth A Neary for her encouragement and best wishes.

Finally, to my two girls, Miss Madelyn E Iannella and Miss Courtney R Iannella, whose many dedicated hours spent watching me write my thesis was an act of true love for which I will always treasure.

I would like to also recognise the once in a lifetime opportunity that Bond University has provided me with. The staff, environment, and participating in the establishment of Australia's first private university will always be remembered.

Abstract

Electronic mail is undoubtedly one of the most successful computer applications in the Information Technology industry. International standards, such as the X.400 Message Handling Systems, have been developed to enable the continual expansion of electronic mail services to a wide range of users. The X.400 standard has comprehensively detailed the functional level of message handling systems but, as with all CCITT standards, has stopped short of defining any requirements for the user interface.

This thesis presents a reference model for the design of graphical user interfaces for X.400 compliant user agents. The reference model was developed from an prototypical implementation of an X.400 user agent called 'Iris'. The Iris reference model was the result of a comprehensive usability study of the graphical user interface for the Iris user agent.

The X.400 issues focused upon in the Iris reference model include the complexity of specifying message recipient addresses and management of a large number of message options. The graphical user interface issues include large list management and forms-based screen design.

One of the most promising international standards is the X.500 Directory Services. X.500 provides a global distributed database of information on people and an extensible range of other objects. The directory enables users to store their complex X.400 addresses and provides a more consistent and reliable method to access this information. The reference model also describes the requirements for the integration of X.500 Directory Services with X.400 Message Handling Systems. This tightly coupled integration is considered to be a powerful enabling technology for electronic mail users.

The Iris reference model provides the computing science discipline with a proven and functional specification for graphical user interfaces to X.400 user agents with X.500 directory support. The processes in developing the model were based on sound design principles backed up by extensive usability trials.

Iris—Messenger of the Gods

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

List of Figures

List of Figures

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

This thesis investigates the design and evaluation of graphical user interfaces for X.400 Message Handling Systems (MHS). The X.400 MHS international standard is a complex and encompassing protocol describing the interchange of electronic mail messages between cooperating systems. One of the key aspects of the X.400 standard is that it does not provide any guidelines for the user interfaces—called User Agents—to electronic mail systems utilising the standard. The research aims to develop a reference model for a graphical User Agent (UA) to a typical X.400 MHS application. The demonstrable model incorporates an adaptive and extensible graphical user interface based upon an open-architecture. The prototype UA, called Iris[1], reflects an instantiation of this model developed for one specific architecture. The scalable model also provides a reference point and criteria for similar application domains.

Another key standard utilised by Iris is the X.500 Directory Services (DS). The X.500 DS international standard protocol describes the structure and contents of a highly distributed database. The database contains information on a large range of objects (eg organisations and persons) as well as enabling new types of object schemas to be defined. The Directory of cooperating agents enable searching of relevant entries in the global database and retrieving attribute information about the objects. One of the original motivations for X.500 DS was to cater for the proliferation of complex addressing structures used in X.400 MHS.

This work involves research into three key areas of information technology, with an emphasis on user interface design:

- Human-Computer Interaction (user interface design and evaluation)
- Open System Interconnection (X.400 MHS and X.500 DS)
- Software Engineering (development tools, environment, and integration)

---

1. Iris is a Greek mythological character who was one of the messenger of the gods and who personified the rainbow.

This research work was partially funded by a CSIRO Collaborative Research Programme in Information Technology. The project is entitled 'Generic Human-Computer Interaction for Electronic Mail', CSIRO grant number 88-13, and headed by Professor Michael J Rees at Bond University.

1.1.1 Human-Computer Interaction

The development of computing software systems involves considerable effort on establishing a consistent and functional user interface. The Human-Computer Interaction (HCI) discipline integrates a multitude of areas that must be considered and studied for the effective design of such interfaces. Failure to do so will inevitably lead to low user productivity of a system as it will inhibit and discourage the sophisticated user population.

HCI does not purport to solve user interface design problems with any magic formula. Instead it provides the background and the impetus for increased attention to the user problems faced with using software interfaces. HCI outlines the potential problems faced in user interface design and offers a rich and comprehensive set of guidelines to counter such situations. HCI does not limit the design of innovative new interface paradigms and encourages experimentation of such design to further empower the user. An aspect of HCI that is becoming pivotal is the evaluation of the user interface designs. Following design guidelines does not guarantee that the final product will have an infallible user interface and most methodologies do not provide such a guarantee. It is for this reason that the evaluation process must be employed to assess and confirm the original user interface design.

The most appropriate method for the development of a user-centred interface for Iris is to use a prototype and evaluation cycle. The development of prototypes enables the interface to be generated in less time and allows feedback to be incorporated into subsequent redesigns. The prototype may be less functional at the application level, but provides the complete interface at the user level.

Prototyping can be achieved using an User Interface Management System (UIMS). A UIMS allows for the management of the total dynamic behaviour of the interface and separates it from the underlying application. A graphical UIMS also provides a simulation of the interface in real-time. Effective use of UIMS tools can dramatically decrease the development cycle and provide a stable foundation to design graphical user interfaces.

By using iterative interface evaluation, the user interface can be refined in a step-wise fashion without leading to any dramatic large-scale changes. Evaluating interfaces is paramount to the successful design. Interfaces designed by programmers are notoriously unusable by the wider user population. Design should involve potential users from different classes such as management, administration, and researchers. HCI defines different interface evaluation methods that can be utilised to provide feedback for redesign of the prototype or application.

The design of the Iris UA includes the following steps:

1 paper-based design
2 evaluation
3 prototype with a UIMS
4 evaluation

The development iteratively cycles through steps one and two, then three and four, to maximise the effectiveness of the interface designs.

1.1.2 Open Systems Interconnection

The CCITT X.400 Series of Recommendations on Message Handling Systems and X.500 Series of Recommendations on Directory Services are the first international standards in the Application Layer of the OSI Reference Model. The major achievement is that they distinguish between the underlying message transfer/database services and the user interface layer (or User Agent). The standards make no provisions for the interface definition of the User Agent which gives enormous potential for developers in designing the user interface. On the other hand, it may give too much flexibility resulting in inconsistent interfaces.

Unfortunately the X.400 standard is complex and can present the user with a deluge of information for both input and output. Research and development of User Agents to X.400 have mainly consisted of character based screen designs where the user is required to specify full options for message transactions (Kairi & Barnard, 1988).

The X.400 standard provides a large range of options for message handling. As apposed to current de facto standard electronic mail systems where the number of options is small, X.400 defines a multitude of service elements in order to support a wide range of messaging requirements. These options include support for multimedia messages, that is messages containing graphics, images, sound, and facsimile parts. X.400 is also extensible to teletext and non-electronic UAs such as the physical delivery postal service.

Being such a wide-ranging and complex standard, X.400 employs a similarly complex addressing scheme for message recipients. X.400 addresses can contain up to a dozen attributes to successfully identify a message recipient. The X.400 address has an attribute/value structure as opposed to the common single user/domain/site name structure of de facto electronic mail systems. The design of the Iris UA concentrates on this issue as well as other general UA design functionality.

One of the major disadvantages for electronic mail users is that there is no guaranteed method of determining a user's electronic mail address. In particular for X.400 users, this represents a major problem. Fortunately the X.500 Directory Services standard enables a worldwide distributed database of object information to be stored and searched. The objects in the database can be defined to include any acceptable schema including countries, organisations, people, and applications. Objects can then be defined with a series of publicly-known attributes. Each entry in the Directory conforms to a number of requirements to enable the successful searching of the database. In the case of X.400 addresses, a person's unknown address can be systematically searched across local, regional, national, and international directories.

The design of the Iris UA also concentrates on the integration of X.500 Directory Services with message addressing. The success and acceptance of X.400 will depend on this seamless and effective integration. Early work by OSI consultants (Molesworth, 1990) outline the enormous opportunities to be obtained from the integration of X.500 Directory Services lookups for X.400 MHS and the importance of a unified user interface. Current research (Whitten,1993) still indicates an enormous need for X.500 implementations to foster the use of X.400 electronic mail systems.

1.1.3 Software Engineering and HCI

Software engineering has traditionally focused on the underlying architectures involved in software development. Early attempts (Draper & Norman, 1985) to change this view have not always been successful since HCI was in its infancy. However, many now argue that HCI and software engineering are two disciplines that need to readdress their own relative scopes. Cockton *et al* (1992) state that HCI should be owned by the software engineering community and that user interface concerns should be an integral part of the development cycle. Denning & Dargan (1994) outline a spiral model, instead of the traditional waterfall model, that leads to a new architecture for the two disciplines to merge.

Software engineering projects are slowly becoming more supportive of effective designs for the user interface and its critical role in the overall success of the system. Gorny (1991) stresses the necessity of a change of the 'technology-oriented production process approach towards the human-oriented attitude of architects'. Beck & Ziegler (1991) stress that HCI approaches, such as a UIMS and prototyping, are practical methods and tools which increase productivity and, moreover, consider the user's needs and tasks. Emerging international standards[2] are now sympathetic to the 'principles of software ergonomics' by requiring software producers to pay particular consideration to usability. The increased attention and research on HCI issues and their integration with traditional software engineering processes will lead to a greater respect for each other's contribution to the industry.

Developing a prototype application requires consideration of the hardware and software requirements essential to successful implementation. The Iris UA is developed for Unix workstations under the industry standard X Window System environment. This allows for easy porting of Iris to any workstation that can support the X Window System.

One of the leaders of windowing interfaces on such workstations is the Open Software Foundation's (OSF) Motif User Environment. It was seen as important that Iris be developed to conform to such standards. The OSF/Motif User Environment provides a framework for development of applications using its User Interface Language and Toolkit library. One of the strengths of the OSF/Motif Toolkit is the extensibility and sophisticated set of interface objects provided by the Toolkit library. Using OSF/Motif also conveys other benefits such as user-customising of colours, fonts, and internationalisation.

Use of CASE tools such as a UIMS for the Iris prototype is also an important software engineering technique. A UIMS enables the software developer the freedom and benefits of separating the user interface from the application code. A UIMS handles the graphical management of the interface whilst providing full support for communication to lower level application functions. A UIMS may not be seen by mainstream software engineers as a true CASE tool nor part of software engineering. These issues are key to the close cooperation of the two disciplines and should be resolved by evidence of successful interaction.

1.2 Purpose of the Research

The focus of this research integrates the following issues in developing the reference model:

- graphical user agent design for X.400 UAs
- X.400 message addressing

[2.] For example: European Community Council Directive—90/270/EEC

- X.500 integration to X.400 UAs
- large list management
- forms-based interface screens
- UIMS development
- user interface evaluation

These areas present formidable challenges for user interface designers.

Primarily, the design of the Iris UA graphical user interface includes sophisticated tools for the handling of X.400 message services. Using appropriate design tools, methodologies, and a comprehensive evaluation process, Iris provides all the functional requirements for an X.400 UA. The prototype has resulted in novel and innovative concepts in the interface for message addressing which is an inherent problem in X.400 MHS. The complex X.400 address structure will require interfaces not found in most electronic mail systems. In specifying a full X.400 address, the user is faced with supplying up to twelve qualifiers about the recipient. These include; surname, organisation, private domain name, administrative domain name, and country. Again this presents a daunting task for a user. An investigation into how X.400 addresses are presented on business cards and other media is also considered.

Solutions to these and other problems are aided by research into X.500 Directory Services and its integration into X.400 UAs. X.500 is seen as a critical part of X.400 acceptance and the provision of X.500 services in X.400 UAs is almost mandatory. Provision of such services requires considerable effort in design and analysis of the X.500 searching algorithms. All effort must be targeted to empower the user with the ability to quickly and successfully search for potential recipient addresses. The interface for such integration should be as transparent as possible. That is, the user should not be required to understand the technologies underlying X.500.

Another area of research concentrated upon is large list management for messaging applications. X.400 UAs enable the user to manage the storage and retrieval of many messages. The number of messages can be quite dramatic and folders can be utilised to manage the message database. Over time the number of folders may become overwhelming and current user interface tools such as pull down menus make it difficult to handle such large lists.

A common interface area requiring investigation is the provision of forms-based screens. Forms-based screens are utilised for the entry of data that is usually provided on paper-based forms. The mapping of the two—paper and display—requires meticulous attention to provide direct and unambiguous representations. This research investigates the format of the X.400 addressing schemes on both paper (eg business cards) and the corresponding requirements for screen entry and display.

This research also investigates the design and development issues utilising UIMS in the implementation cycle. The benefits of UIMS are reported on and challenged in the design and implementation of a real product. Associated with this is the important area of interface evaluation. This process is investigated and applied to the Iris prototype.

1.3 Scope of the Research

This research investigates the design and implementation of prototype for an X.400 MHS user agent. In doing so, the benefits of X.500 Directory Services will figure prominently to enhance the UA and increase functionality. The UA will follow the interpersonal Message (IPM) Han-

dling System user facilities and include all basic and optional service elements (defined in X.420). The message content will be limited to text-only message body parts.

The Iris UA prototype will be developed with the X Window System and will be based on the Open Software Foundation's Motif User Environment. Implementation of the prototype will involve a structured design and evaluation process. The model design solutions are likely to provide considerable commercial potential for developments into user interfaces for UAs. Exploitation of these solutions will be beneficial to user interface developers.

1.4 Importance of the Research

This research highlights some unique and novel features required for an X.400 UA. These include:

1 A comprehensive and functional reference model that outlines the graphical UA interface requirements for X.400 message handling systems. The Iris UA interface model designs provide a benchmark for the judgment of similar X.400 IPM UAs.

2 X.400 addressing applied to a user tailorable interface for the specification and entry of complex and structured data. This is also significant for other application domains that involve the entry and display of attribute/value structured information with form-based screens. The presentation of X.400 addresses on business cards and other media is also examined.

3 The integration of X.500 Directory Services into X.400 UAs for the purpose of automatic retrieval of recipient addresses. Providing a simple interface to the directory is of paramount importance for the wide acceptance of X.400 MHS.

4 Alternate methods for interaction to large lists of data. In this case, the large number of message folders requires novel interface methods and structures.

5 The design and implementation process utilising the purported benefits of UIMS tools. This includes an comprehensive study of various user interface evaluation techniques.

As an added benefit to the above, particularly points 3 and 4, some aspects of the research can easily be modified for use in non-X.400 electronic mail systems, thus extending the practicality and applicability of the research outcomes.

1.5 Thesis Outline

Each chapter describes a tightly coupled coverage and analysis on an aspect of the research by focusing on the issues raised and the applicability to the project's goals. Later chapters concentrate on the current issues whilst providing insights into the foundations and principles laid down in the earlier chapters.

Chapter 2 on *Human-Computer Interaction* covers the central issues of this multidisciplinary field. Various HCI models, supporting architectures, and the importance of human factors are discussed and outlined. The major section of this chapter deals with the principles and methodologies involved in designing graphical user interfaces and examples are presented. User Interface Management Systems are also discussed and reviewed in relation to the design, prototyping cycle, and development of user interfaces. The benefits of user interface guidelines and standards are outlined as well as the importance of user interface evaluation methodologies.

Chapter 3 on *Message and Directory Standards* introduces the concepts, functionality, and applications of electronic message systems. A discussion of the architecture, standards, addressing, and integration of electronic mail systems is presented. A comprehensive overview of the X.400 Message Handling Systems and X.500 Directory Services standards are outlined.

Chapter 4 on *Related Work* covers the areas of research and development that currently overlap with the work covered by this project. A discussion of the design of electronic mail interfaces and functional needs of electronic mail systems are identified. A review of X.400 user interface design, and issues of X.400 addressing are highlighted including the integration of X.500 Directory Services. A comprehensive review is presented of current graphical user interfaces for both X.400 and non-X.400 electronic mail systems. Finally, a number of X.400 and X.500 and development systems are overviewed as these provided the framework for the implementation of Iris.

Chapter 5 on *Design and Implementation* outlines the influences guiding the development of the Iris UA. The design and subsequent implementation became the basis of the Iris reference model. The strategy involved in the design and early evaluation of the Iris interface is fully discussed including those aspects and deficiencies of a UIMS as a prototyping tool. The X.400 and X.500 underlying support systems were investigated for the prototype as this was necessary for realistic interaction sequences. Message addressing using X.400 addresses and the integration of X.500 Directory Services for address lookup is comprehensively discussed. A new algorithm is proposed which maximises the matching success rate for directory lookups.

Chapter 6 on *Evaluation and Usability* of the Iris interface design discusses the selection of appropriate and effective evaluation methodologies and the scenario-based usability testing process. The usability tests are video-taped for later analysis and the 'thinking-aloud' method is used by the trialist to provide comparison of the user and interface models. A number of questionnaires are discussed and a comparison of recorded times for each task both provide a statistical analysis of the usability trials. The usability evaluation results are then used to identify interface problem areas that require redesign and the entire process is repeated iteratively until adequate positive results are obtained. The evaluation process was instrumental in developing the final designs of the Iris reference model.

Chapter 7 on the *Reference Model* describes in detail the generic graphical user interface elements for X.400 UAs. The reference model details the requirements for the IPM UA service elements with respect to a graphical user interface implementation. The reference model was a result of the design, development, and evaluation of the Iris UA interface outlined in chapters five and six.

Chapter 8 on *Discussion and Conclusion* presents an analysis of the results of the research. The results are compared to the goals and objectives of the original project specification and appropriate conclusions are drawn. A synopsis of the implications of the research is outlined and its relation to the HCI discipline. A brief overview is also presented on the possible future work applicable to the project.

Chapter 2

Human-Computer Interaction

2.1 Introduction

This chapter covers the central issues of Human-Computer Interaction (HCI) and their applicability to today's computing environments. Since HCI is multidisciplinary, researchers in the field have had a arduous time in precisely defining what areas this discipline covers and on a universally agreed definition of the user interface. Although this has not been thoroughly agreed on by practitioners, some, like Shneiderman (1986) have issued challenges for researchers in an attempt to mature the discipline.

HCI is based on sound conceptual models and architectural frameworks. Various HCI models, from structured to object-oriented, are discussed and supporting architectures outlined. HCI involves the study of humans interacting with computer systems and relies on the principles of cognitive psychology in the human factors discipline. The importance and integration of human factors into user interface design is outlined.

The major section of this chapter deals with the principles and methodologies involved in designing user interfaces. In particular, the issues involved with graphical user interfaces are outlined and examples of design considerations are considered. Some examples from professional software applications are critically examined with interface design problem areas highlighted. The advantages of User Interface Management Systems (UIMS) in the design, prototype, and development of user interfaces is discussed. Various UIMS are briefly reviewed and the iterative prototyping development cycle is discussed.

The benefits of user interface guidelines and standards are outlined with a review of some common sets of guidelines. The importance of evaluating user interfaces with real users and the range of techniques available are summarised. Finally, a look at the future for user interfaces is examined with some key HCI researchers postulating the research and development interface issues of tomorrow.

2.2 HCI Concepts

In their classic paper, Hartson & Hix (1989a) provide a comprehensive survey of the concepts of HCI. These concepts provide the underlying framework of the HCI discipline and include:

- Dialogue Independence
- Structural Modelling of the Human-Computer Interface
- Representation of the Human-Computer Interface
- Interactive Tools for Human-Computer Interface Development
- Rapid Prototyping
- Methodologies for Human-Computer Interface Management
- Control Structures for Human-Computer Interface Management

Dialogue independence is the separation of the user interface of a system from the application-specific code. This is an important area in which User Interface Management Systems (UIMS) are providing support (see '2.6 User Interface Management Systems' on page 30). This separation allows for greater flexibility in changing the interface without modifying the underlying application architecture.

Structural modelling describes the general processes involved in HCI. This includes the structure of the interaction between end-user and computer and identifying interface objects that the designer requires. Representation of the HCI involves recording the interface design in some form of metalanguage or formalisation technique. With synchronous dialogue, representation could be recorded using BNF notation or state-transition diagram. With asynchronous dialogue, an event-based mechanism is used.

Interactive tools refer to software tools to develop the interface from the representation. The tools can usually also be used to evaluate, prototype, and implement the final interface. Rapid prototyping refers to the process of iterative refinement of the interface. This involves development of a interface that is tested with end-users and any feedback is used to develop a refined interface. This is again tested and improved until the end-user and developer are satisfied. Since only the interface is developed, the process is quicker and leads to a more acceptable end product.

Methodologies are management procedures to incorporate the interface development into the entire lifecycle of the project. Traditionally the interface was developed last and was not regarded as being part of the lifecycle. This, of course, has dramatically changed in the last few years. Control structures reflect the sequence of interactive steps in a system that can influence the way it is designed and implemented. Again, synchronous and asynchronous dialogue techniques are involved.

Attempts at a taxonomy of HCI terminology (Chignell, 1990) have provided firm foundations for HCI. Chignell separates HCI into four general branches:

- Basic Interface Model
- Cognitive Engineering
- User Interface Engineering
- Applications

Each branch is further broken down into numerous sub-branches and sub-topics.

After reviewing over 180 research papers in the area of HCI, a suitable expansion of this list can be made:

- Models and Architectures
- Human Factors
- Interface Design
- User Interface Management Systems
- Guidelines and Standards
- Evaluation and Usability

Each of these branches are also broken down into subsections and reflect an expansion of the HCI discipline.

2.3 Models and Architectures

Like any discipline, HCI requires models and architectures as a basis for developing interface paradigms. These provide a basis to develop consistent and more sophisticated interfaces. Interfaces that are based on fundamental and proven models and developed with underlying architectures are a firm foundation for developing successful interfaces. Building mental models allow designers to be more competent and in control of software development (Potosnak, Sept 1989).

2.3.1 Models

Many models exist that have been developed a number of years ago that are still useful today to describe the overall *concept* of HCI. Some of these models are unable to describe current interface models that are interactive and rely heavily on a graphical interface, others have been expanded to provide such support. One of the first and best known of these models is the Seeheim Model (Green, 1986) which divides the user interface into three distinct components (see Figure 1).

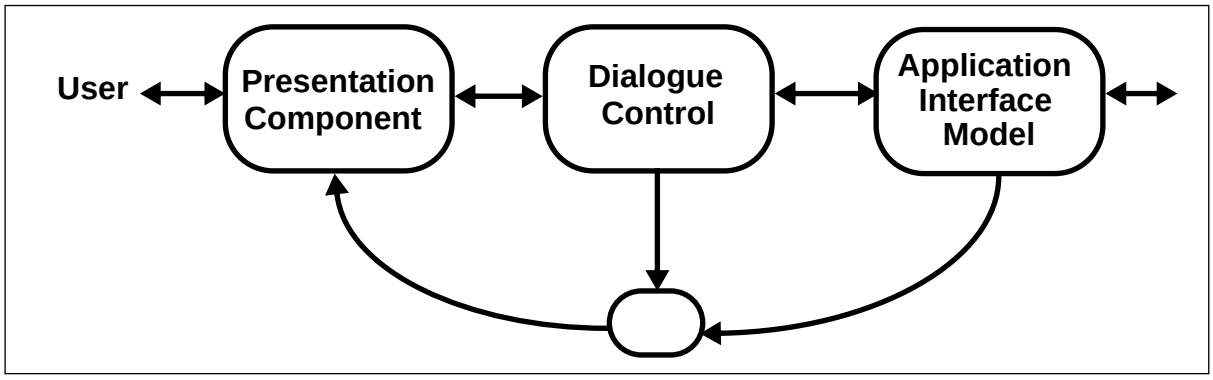


Figure 1: The Seeheim Model

The presentation component deals directly with the physical devices of the user interface including input and output devices, screen layout, and display techniques. The dialogue control component deals with the structure of the commands used between the user and the computer. The application interface model deals with the connection between the user interface and the rest of the application.

Using the Seeheim model, the application can change the state of the user interface by sending *tokens* between the components. The tokens consists of appropriate data values and is used to

transfer control information between the user interface and the application. In doing this, the user interface does not need to communicate with the application code to determine the state of the user interface.

This apparent separation of the interface from the application is extremely important and features in almost all HCI models. The problem then arises as to where to draw the line between interface dialogue and application (Hartson, 1989). Further work has been attempted on modifications to the Seeheim model (Edmonds, 1990) in an attempt to emphasise the separation of interface from application. These mainly involve slight changes and additions to the Seeheim model, indicating its acceptance as an important HCI model. The Seeheim model has also been used as the basis for some commercial graphical User Interface Management Systems such as TeleUSE[1]. Models for graphical interfaces (Hutt &Flower, 1990; Farber, 1989) are extensions to the Seeheim model and simply provide support for the graphical development environment usually via extra components. Most of these models concentrate on the output of text and graphics to the computer screen.

On the other side of HCI is the input that the (human) user generates. Myers (1990a) has developed a comprehensive model explicitly for input into a graphical interface. Part of his Garnet project used a model called *Interactors* which allowed the developer independence from the graphical window manager, yet allowing support for all conventional input (mouse, keyboard etc). The Interactor handles communications between the input device and the graphic elements on the screen (eg menus). Building user models can also be used to design user interface components. Barfield (1993) describes the issues involved of users moving from text-based to menu-based systems and the problems involved in redesigning the menus to meet with the user's models. In this case, cascading octagonal menus were designed to maximise visual feedback and minimise 'muscle memory' (hand-eye coordination).

Object-Oriented Models

As the computing industry becomes aware of the advantages of developing software using object-oriented programming languages, we are seeing the emergence of HCI models utilising this new paradigm. Such models (Hurley & Sibert, 1989; Zainlinger, 1990; Zhou & Kubitz, 1992) describe objects and the operations that can be performed on them. The object-oriented principles of reusability and inheritance can be directly related to user interface objects. For example, items in a pulldown menu would inherit the resources of the parent class including methods to display the item (see Figure 2).

The object-oriented models provide a new and extensible system for the development of user interfaces. What they do lack is a *uniform* and standard environment for development. Industry standard interface toolkits need to be developed to fully exploit these models. Attempts have been made (Fekete, 1991; Backer & Genau, 1992; Fischer, 1987) but these have not become de facto standards as they are usually built on top of a non-object-oriented system. A successful example of an object-oriented user interface toolkit is InterViews (Linton *et al*, 1989). The InterViews toolkit offers a rich set of predefined interface objects and composition mechanisms. The InterViews composition facility allows the developer to combine interface elements and concentrate on the higher level semantics and behaviour of the newly defined interface object.

---

1. TeleUSE is a registered trademark of Alsys

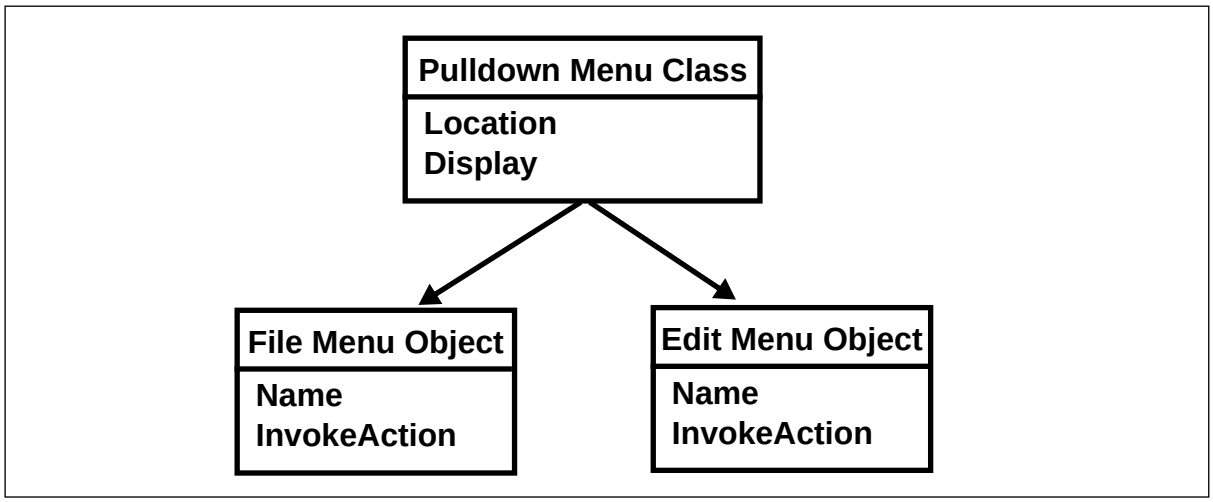


Figure 2: Object-Oriented Model Example

2.3.2 Architectures

User interface architectures define the framework and structures in developing the interface for applications. Lane (1990a) defines a set of rules and dimensions that define the structural application model. Such guidelines provide a more detailed environment than interface models and are more explicit in the development of the user interface. Lane later defines a methodology to reuse the structural designs in previous user interfaces with known solutions and known properties (Lane, 1990b). Others (Sonnenwald, 1990) define architectures that are specific to human interaction and computer networks.

The Picasso application framework (Rowe *et al*, 1990) is an example of a system that includes a toolkit with an architecture. The architecture provides high-level abstractions to support the development of graphical user interfaces and the toolkit contains a library of interface objects. It is also an example of an object-oriented architecture which encourages the design of reusable interface abstractions. Other architectures have been defined on the basis of the application domain. Navarro (1990) proposes an advanced user interface architecture framework for group communication activities based on the integration of object-orientation, multimedia, intelligent interfaces, and user interface languages.

2.4 Human Factors

One of the multiple disciplines in HCI is the area of human factors, which is closely related to cognitive psychology. Cognitive psychology investigates how the human brain thinks in various situations. With this knowledge of how the brain forms mental models, designers could represent and map this behaviour in the computer user interface. As practitioners are yet to completely solve this problem, they resort to designing models of the brain and attempt to match it with a model of computer interfaces. The user will develop a mental model of the system that the designer wants to be compatible with the underlying conceptual (design) model (Norman, Donald, 1986). If the *user's model* and the *designer's model* differ in any respect, then the user will experience some level of problems when interacting with the software (the system image). Norman's model is graphically shown in Figure 3. The use of models in the HCI field is rather complex with a wide range available (Nielsen, 1990c),

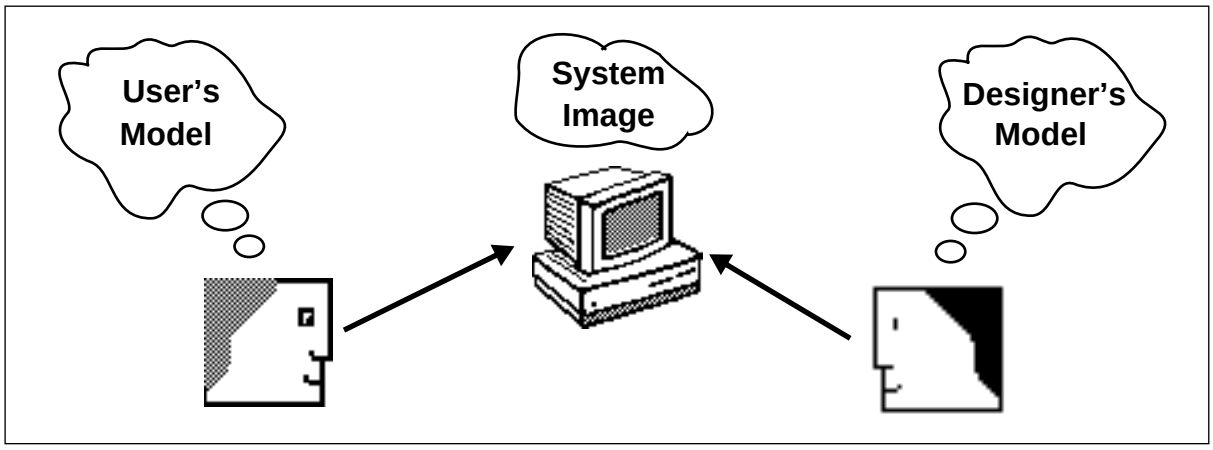


Figure 3: User and Designer Models

Donald Norman (1986) has developed a seven stage model that represents the activities that occur when users perform a task. This is shown in the first column of Table 1 with the second column giving an example of each activity.

Table 1: Norman's Stages of Activities

Stages of Activity	Example
Establish the Goal	Make a sales chart more 'dramatic' in a report
Forming the Intention	Change the scale on the sales axis to start from $10,000
Specify Action Sequence	In the spreadsheet application, use the Format Chart dialogue
Executing the Action	Select the Format Chart from Format menu and enter 10,000 as the x-axis starting value
Perceiving System State	Did the chart's axis change?
Interpreting the State	Did they change by the specified amount?
Evaluating System State (with respect to the original goals)	Does the chart look more 'dramatic'?

From Table 1, we see that there are many activities that the user performs (some subconsciously). The problem arises in the *gulf* between the users goal and the execution required (called the 'gulf of execution') and between the original goals and interpreting the systems current state (the 'gulf of evaluation'). This is graphically shown in Figure 4.

Clearly, having large gulfs of execution and evaluation will inhibit use of the computer system as the gap needs to be bridged or made narrower. Norman describes two ways to overcome this:

1 move the user closer to the system
2 move the system closer to the user

The latter is the preferred option as it does not require the user to change their conceptual work habits. This does however involve extra development in establishing exactly how users conceptualise their activities and designing an interface to map directly to this model.

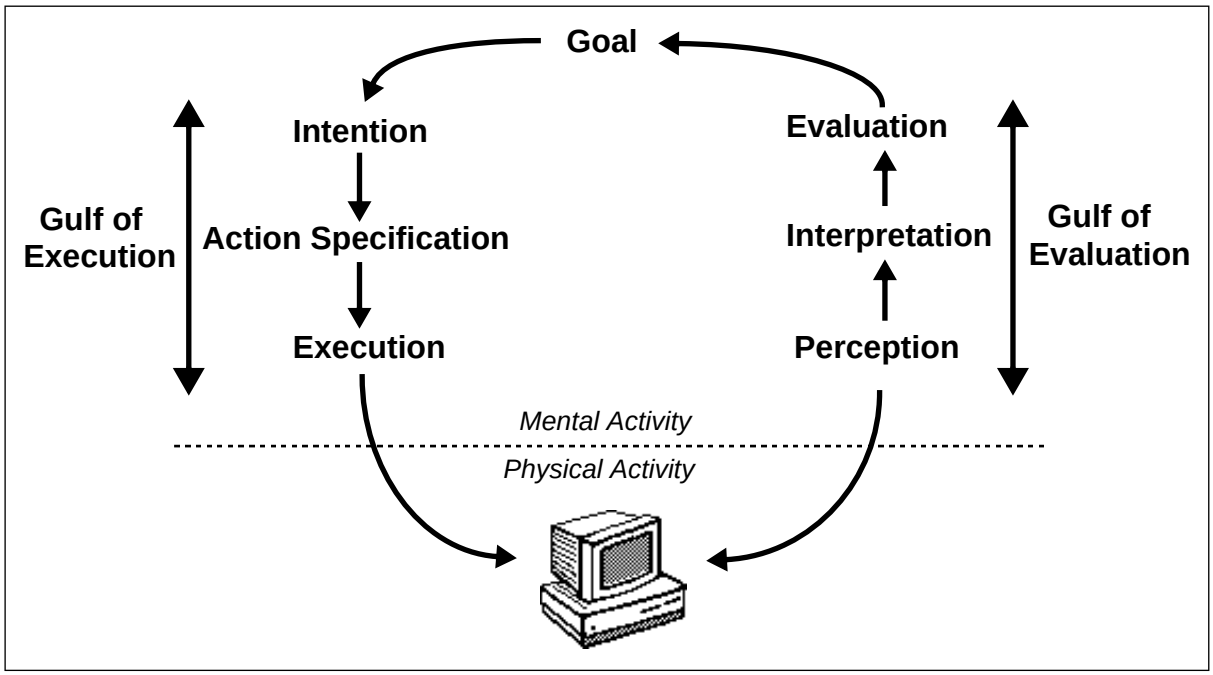


Figure 4: Norman's Activities Model

Artificial devices that are designed to display or operate on information enabling a clear representation can be used to aid in the above situation by enhancing the human's ability. These are called 'artifacts'. An example of the use of an artifact would be in the use of legends for charts. Figure 5 shows an unnatural mapping (top) and a natural mapping (bottom) for a chart legend representing percentages (which uses an additive scale). It can be clearly seen which legend is the better representation with darker shades mapping to higher values. This may seem trivial, but this cognitive artifact is an example of successful mappings between the user and the system model. Donald Norman (1991) describes how these artifacts pervade our every day lives and Thomas & Kellogg (1989) describes ways in minimising these artifact gaps in interface design.

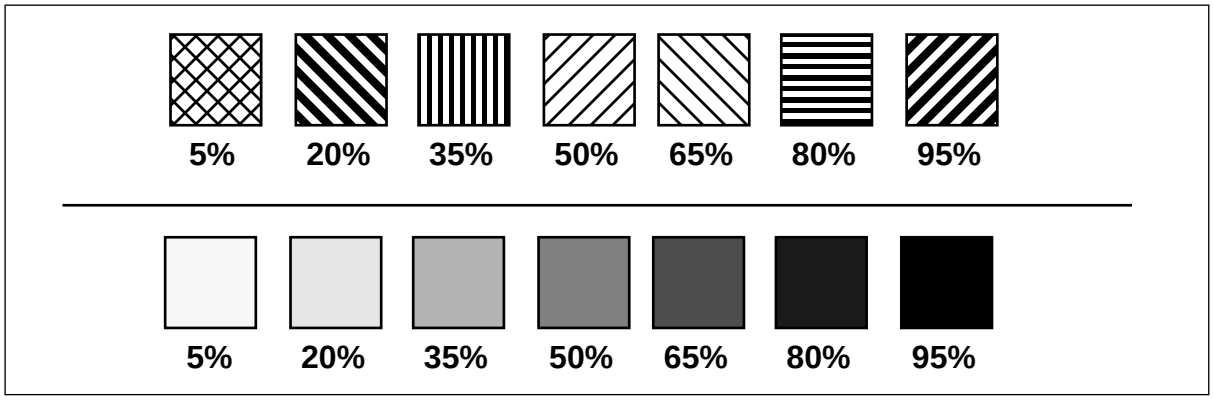


Figure 5: Artifacts Example

Human factors is a broad area that covers not only HCI but the also the social sciences (Anderson *et al*, 1991) and how social factors affects the design of products. The classic example is that of programming a video recorder and the difficulties involved in mastering the timing mechanisms of modern VCRs. The designers of such systems apparently have not taken into account the usability of such devices and have not even attempted to provide an 'easy to use'

interface. The design has not changed over the years even though many users have voiced their dissatisfaction (Thimbleby, 1991).

Successful design of any product or system relies on the close cooperation of the design team and the human factors specialist. Such a specialist can increase the utility and make quality and reliability perceptible in computer interfaces (Schaffrina, 1991).

2.4.1 Design and Integration

Human Factors design can also be applied to specific areas of interfaces such as menu selection (Kent Norman, 1991) and interfaces for multiple windowing systems (Kent Norman *et al*, 1986). One of the more popular areas is in designing for human error. That is, designing systems that are aware of the possible errors of humans. After all, humans do make mistakes and designers should assume that they will do so using their software. By analysing human errors and classifying appropriately, Donald Norman (1983, 1990) has developed a set of guidelines to overcome this possibility (such as feedback and consistency). Further work by Lewis & Norman (1986) on minimising and detecting errors conclude that systems should aid the user when error situations occur by providing clear explanations and providing suggestions for recovery.

Integrating human factors into the design cycle has proved very successful (Gould *et al*, 1990) and methods of integrating human factors into the development cycle are beginning to appear (Lund & Tschirgi, 1991; Whitefield & Sutcliffe, 1992). Many large software organisations currently use and have been using human factors in contributing to the development of products (Day, 1989). Studies have shown that by incorporating human factors into the software lifecycle, substantial financial savings occur as the maintenance and training costs decrease (Mantei & Teorey, 1988). Eason & Harker (1988) outline the human and organisational issues that are obstacles to the integration of human factors into current information technology systems. They conclude that new technical capability will not be utilised if human factors strategies are not pursued. Rubin (1986) also has similar conclusions and believes that human factors is the axis upon which the next generation of communication systems hinges.

Human factors provides the necessary underlying cognitive models that allow designers to implement interfaces based on practical analysis of human tasks and modelling human thought processes. To be successful in utilising human factors in a project, development teams rely heavily on management allocating resources and changing the design process to an iterative process (Potosnak, March 1989). This includes specifying concrete goals for producing better software that go beyond broad claims such as ‘easy to use’ (Potosnak, March 1988). Management must also include real end-users in the design process, not to do the design but to provide feedback to interface options (Potosnak, July 1988). Finally, human factors can coordinate the modular implementation (separating the interface from the application) which benefits application designers, programmers, and users (Potosnak, May 1989).

2.5 Interface Design

The design of a user interface is the crux of the development of any software system. In recent years, software projects have been devoting more person-power to the design of the interface than the underlying application code. This may simply be a sign of a competitive market, where the users are demanding more sophisticated software interfaces. Either way, this is a promising sign for the industry.

There are many reasons to apply appropriate design criteria to interfaces. These range from interfaces that simply annoy users (such as continually setting default values in a dialogue) to fatal instances (Neumann, 1989). A discussion of user interface design and software engineering can be found in (Iannella, 1991) and (Draper & Norman, 1985). There has been no single user interface design methodology that has evolved as the de facto standard. Many theses has been written solely on the design of human-computer interfaces (eg Arbour, 1990) but usually a knowledge and application of various methods is required.

There are generally two solutions (Good *et al*, 1984) to the user interface design problem (based on Donald Norman's earlier work). The first is to adapt the user to the system which involves changing the user's normal cognitive behaviour, extensive training, and documentation and should be avoided. The preferred solution is to adapt the system to the user and can be achieved by following user interface design principles.

2.5.1 Principles

Laurel (1986) describes a user interface as being 'mimetic'. That is, it presents a particular kind of representation to the user that has the following features:

- arranged in harmonious or pleasing way,
- acted out rather than described, and
- represents interaction objects.

The idea behind this is that designers can use experiences that users want to have with interactive representations, given the constraints of current technology.

To leave the design of an interface to a designers experiences and intuition may lead to problems with end users. One of the major principles of user interface design is to test design prototypes with users and to incorporate any feedback into new designs. (See '2.6.2 Prototyping' on page 35.) There are many advantages to this design cycle resulting in better interfaces (Myers, 1989):

- rapidly developed design
- ease of change
- consistent interfaces (provided by the prototyping tool)
- involvement of specialists (eg human factors, graphic artists, users)
- separation of application code from interface code
- maintainable and reusable code

Another underlying theme for user interface design is consistency in three areas:

1 Internal—eg consistent design and position for screen objects
2 External—eg consistent icons between applications of same nature
3 Real-World—eg traffic stop sign to represent an alert icon

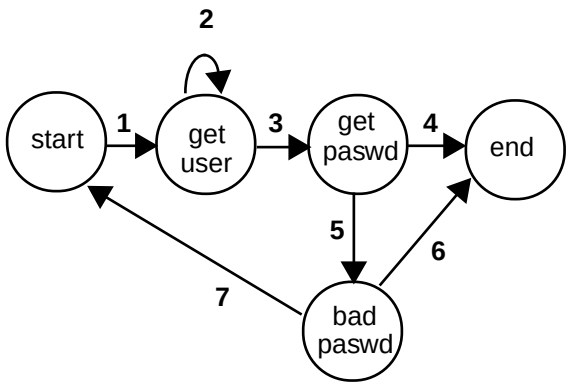
An effective user interface should provide a visual communication tool for the user. The user interface should be comfortable to use and also innovative by providing a beneficial (eg assisting in a user's task) and customisable environment. Customisable interfaces are also a key principle in the design of user interfaces. At some stage in the use of the software system, the end-users will become increasingly proficient and will be hindered with 'too-helpful' interfaces. For example, interfaces that continually prompt the user for confirmation of actions. The user must be able to customise the level of attention from a system, and hence, indicate the

level of assistance required. The interface should provide mechanisms to customise some of the more personal needs of the end-user. For example, the colours of the windows, or the typefaces used for text entry and display.

2.5.2 Formal Methods

Formal methods can be used to describe the properties of a user interface in such a way as to unambiguously represent the tasks, dialogues, and high level requirements between interactive systems and the user. The formal methods do not usually specify the internal implementations. Attempts have been reported on formal methods using the Z notation (Abowd *et al*, 1989) that successfully represent underlying information systems. Formal methods in Z usually require abstract mathematical models to describe such interactive systems. This is one of the reasons why such methods are not common in development and are usually applicable to pure research projects.

Early techniques using State Transition Diagrams (STD) and the Backus-Naur Form (BNF) have also been successfully used (Jacob, 1983) and later refined for hybrid interfaces (Gerlach & Kuo, 1991). Figure 6 shows an example STD and specification of a user 'login' to a system.



1—Print "Enter Name"
2—If not valid user, print "Incorrect user name, re-enter"
3—If valid user, print "Enter password"
4—If password ok, login successful
5—If password not ok, print "Invalid password, re-enter"
6—If password ok, login successful
7—If password not ok, print "Invalid password, start again"

Figure 6: Example State Transition Diagram

The STD notation has difficulty with highly manipulative interfaces but does allow the designer to concentrate on the required system semantics.

Formal techniques for user interfaces has also instigated the development of specific formalisation languages for computer system interfaces. Foley *et al* (1989) describe an Interface Definition Language (IDL) that allows the designer to work at a higher level of abstraction. IDL uses an object-based syntax that can describe interface objects as well as a knowledge base that covers the conceptual design of the interface. It is a high-level description of the interface and as a result, requires further design before implementation.

Extensions to common programming languages, such as Pascal, have also been proposed to capture the user interface (Lafuente, 1992). Foley (1987) developed a formal representation for key aspects of the conceptual and semantic design of interfaces with the goal that this could be directly used by a User Interface Management System as input. (See '2.6 User Interface Management Systems' on page 30.)

2.5.3 Methodologies

There are many methods that support the design of the user interface and no single design method is 'all-encompassing' in that it can be applied to all types of interfaces. Some of these methods have been developed based only on the designer's or programmer's expertise and may not reflect the complete requirements of an interface. It is therefore imperative that methods apply the following conclusion (Bodker, 1991):

> *'Good design methods must prescribe that the means applied in a specific design activity must originate from the use activity in question.'* (sic)

Designers may need to use a hybrid mix of methods to develop the interface by applying each sub-method to specific parts of the system. For example, some initial designs on paper may be a sufficient starting place. This could include a series of 'walk-throughs' with the end users. The designers would elicit feedback as the end user is shown through the paper-based interfaces. After this, a computer generated prototype may be utilised to further refine the interface, and feedback from the end-user can be used in subsequent enhancements. This prototype may include the use of the interface toolkit that would be used on the final system—giving a clearer and more realistic feeling to the end-user.

Finally, common 'toolkit-dependent' guidelines or industry based user interface guidelines can be used to supplement changes to the interfaces. Such guidelines may simply recommend which colours to use in certain situations or may force more dramatic changes.

It is important to use a method that is as general as possible in the early stages of the design process and then refine this later and use more specific methods as the development progresses. At all stages the end users (or some form of human factors design) should be involved, particularly if the designers are not familiar with the domain of the system being designed. Methods that utilise diagrams to represent the user interface have also been developed. Such methods as Rovira Diagrams (Rovira, 1990) use a simple set of symbols to represent the dialogue's syntax and describe the interaction in a consistent manner.

Another type of method is one in which an organisation provides the framework and infrastructure to design systems for particular hardware and software environments. One example of this is IBM's Systems Application Architecture (SAA) which provides an extensible and independent methodology for user interface design. This method has previously undergone testing for human factors and user satisfaction, thus providing consistent interfaces across SAA applications (Dunfee *et al*, 1988; Uhlir, 1988). SAA has been significantly improved by enforcing the Common User Access (CUA) user interface guidelines (see 'Common User Access Guidelines' on page 40) and the separation of the application logic from the management of the displays (Artim *et al*, 1990).

2.5.4 Graphical User Interfaces

The first well known computer software to use a graphical user interface (GUI) was developed in 1962 and called Sketchpad[2]. The application allowed the user to draw points, lines, and arcs on a display with a light pen. The objects could be moved around the screen and constraints and relationships between objects could also be assigned. Considering the state of hardware at the time, this was an revolutionary step for GUI's. Since then the GUI has evolved to include

[2.] A PhD thesis by Ivan E Sutherland at the Massachusetts Institute of Technology

more sophisticated windows, icons, different styles of menus, and various input devices (eg mouse, tablet). A history of the GUI can be found in (Perry & Voelcker, 1989).

With the maturing of the industry comes design recommendations for specific software systems and generalised GUI guidelines. Both types of design recommendations should be followed as the former will result in consistent software across each platform and the latter will result in consistent interfaces across all platforms. A comprehensive review of the different types of GUI's across the different platforms can be found in (Marcus, 1992; Taylor, 1991; Australian Personal Computer, 1990). Specific guidelines for Microsoft Windows can be found in Capucciati (1993), Brackeen *et al*, (1989) describes the Macintosh interface, and Marcus (1990) outlines general interface guidelines.

Hayes (1992) describes the benefits that graphical user interfaces has brought to the Toronto Stock Exchange (TSE). Up to $1 million in stock changes hands every minute and productivity is dependent on how fast a trader can register a buy or sell order. The TSE replaced all the text screens with touch screen terminals running X Windows. This allowed the traders to increase productivity and did not require significant amounts of training to use the system. In other cases, one graphical screen was used to replace three text screens with the same advantages. In some cases, the benefits of the GUI have also been transferred to text-based systems with tools that apply a GUI layer over the command-line interface. Hesketh (1991) describes one such system in which graphical buttons are applied to Unix command scripts on X Window workstations.

The impact of the design of GUI's over text based software is considerable. The following sections detail some of these considerations and examples.

Screen Layout and Windows

The overall layout of the screens should be the first aspect of the design process. Early prototyping with screen layout on paper and experimenting with different layout configurations will lead to a consistent and functional design. The design of the screen layout also includes analysis for the following:

- Functional areas (eg titles, help text, buttons) should be agreed on and consistently followed throughout all the screen displays.
- Separate information on the screen using lines and boxes to group and emphasis critical information components. Liberal use of white space is an effective screen layout tool.
- Ensure that textual information to be displayed on the screen is readable by using appropriate typestyles and is typeset consistently. It is not recommended to use more than three different typefaces and point sizes on the same screen. A mixture of upper and lowercase characters is also more readable than all uppercase characters.

The window design environment has evolved into a set of industry and defacto standards by software companies (Apple Computer, Microsoft) and industry leaders (Open Software Foundation, Unix International). The design and manipulation of these windowing systems (via title bars, close boxes, and scroll bars) have been defined and tested and only in extreme situations should these systems be changed to suit highly complicated interfaces to specialised software systems.

Graphical window environments depend on the mouse as the most common input device which is equipped with between one and three buttons. These buttons have been assigned de

facto standards. In the X Window environment, the left button is used for selection of text (copy) and the middle button for pasting. In the Macintosh environment, the user is fixed with one button for selecting text only. Assigning different functions to these buttons in the manipulation of windows and graphical interfaces should be avoided.

Menus and List Selection

Menu design involves the appropriate arrangement of functions into each menu list and similar options being grouped together. Research into the usability of menu systems and the most appropriate ways of representing menu options (eg alphabetically and numerically) reveals that users generally do not remember the details of the interface. Users may have better semantic knowledge than lexical knowledge (Luff & Heath, 1991). That is, the user knows that the functions exists but does not necessarily know the names of the functions in the menus.

Menus can be categorised into three main types:

1 Pulldown menus which are usually visible on the top of the application widow.
2 Popup menus which are usually accessible only in certain areas of a window.
3 Option menus which display a selection of options to chose from.

The first two types may also incorporate a hierarchical system in which a menu item generates another full menu system. Irrespective of the type of menu system, an acceptably designed menu system should include considerations for the following:

- Feedback to indicate to the user which option is currently selected. This is normally performed by displaying a check mark (eg ✓ or ◆) next to the item name in the menu list.
- Keyboard short-cuts (accelerators) should be provided on most (if not all) of the menu selections. This is useful for experienced users who prefer to use the keyboard instead of the mouse for menu selections. The keyboard accelerator sequence should be displayed next to the item name in the menu list.
- Feedback to the users as to the state of disabled menu options (ie 'greying' out the words is a common example). Removing the option from a menu is not recommended, as this will be inconsistent for the user.

An example of inadequate feedback of the existence and accessibility of option menus can be seen in Figure 7. This was a problem noticed by Apple Computer (Jenson & Sullivan, 1990) in which it was not clearly obvious by the shadowed box alone that an option menu existed. A new style was introduced that included a downward pointing arrow.

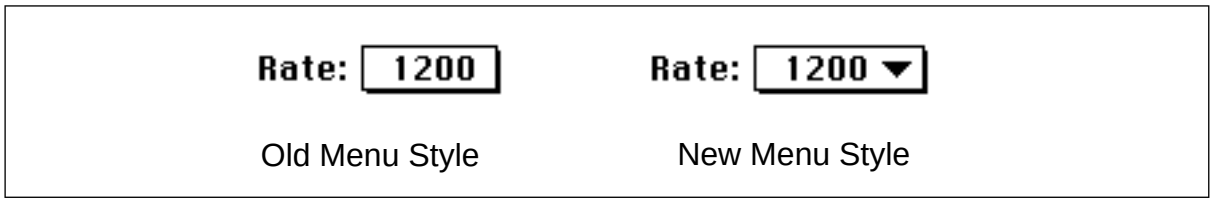


Figure 7: Apple's Option Menu Changes

Providing a list to users to make selections is both an effective interface component and reduces the chances of errors. This is exemplified when the user has to fill in some type of form with data from the real world. If the system expects a finite number of options, then these options should be provided in a convenient list for the user to select from. This will ensure that the response is always correct (minimal error checking) and provides a faster data entry method for the user.

Figure 8 shows an example in which the user is asked to select the number of degrees an object should be rotated. In this case, an option menu of the numbers 1 to 360 is too excessive, so a text field is provided for direct entry as well as a menu of the more common angles.

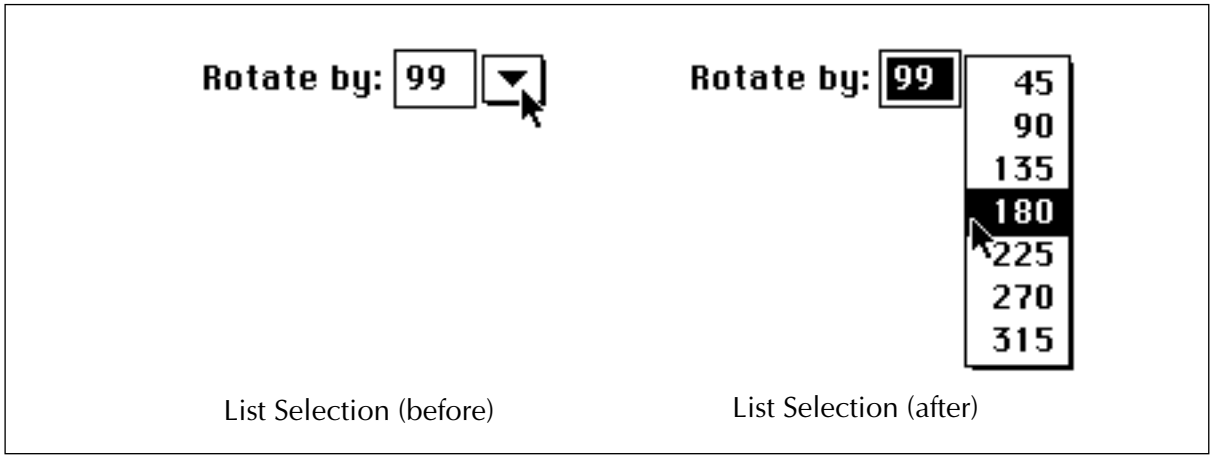


Figure 8: List Selection Example

Research into better visualisation techniques for lists has resulted in attempts to add attribute specific information about the list items to the scrolling mechanism. Chimera (1991) uses the concept of 'value bars' that reflect the size or modification date of a file directory system. The value bars assigns relatively sized regions near the scrollbar according to the item's attribute value. In the case of a file directory, the size of each file may be reflected in the value bar, and the user can thus perceive the overall distribution of the items in the list. Liao *et al* (1992) describe the evaluation of three different methods of file directory lists. Using list item indicators, different colours, and selective list display resulted in no clear advantage but the latter is more effective for very long lists.

Menus and finite list selection methods have proven to be essential in modern graphical user interfaces. The use of such interface techniques is frequently used in the Iris prototype as their utility is beneficial when consistently used by end users.

Icons

Icons are small graphic representations of some function or object. Icons can be used to set up an analogy between processes and user tasks and usually give visual clues to the user as to the functions they perform. For example, a paint brush icon in a graphic program should allow the user to freehand paint on the application window.

Two advantages of using icons are that they can be more universal (across cultural groups) and can be recognised more rapidly (Rogers, 1989). The icon set should have a design theme (consistent look throughout the system) and conform to de facto industry standards (eg rubbish can for deletion).

Copious use of icons can lead to user memory-overload problems and severely hamper the interface. Appropriate feedback should be provided to indicate each icon's function if the numbers are large or possibly unclear. See section 'Examples' on page 25 for a discussion of feedback for icon usage. It is sometimes useful to include a textual name under each icon to reinforce its function but this may lead to translation problems. Icons can also present problems with cultures that may have different meanings or unknown usages of certain icon representations. An example is outlined in section 'Internationalisation' on page 24.

Only recently have draft standards on icons begun to appear. These standards attempt to present a series of graphical icons that follow a set metaphor and contain rules on the representation of the objects and actions. The standards also define the functionality of the symbols in the set and the recommendations for the interactive properties of the icons. As an example, the ISO/IEC draft standard (ISO/IEC, 1992) on 'Graphical Symbols Used on Screens: Interactive Icons' provides common representations and interactive rules for desktop objects such as printers, documents, folders, and rubbish cans.

Although the use of icons is important to graphical user interfaces, they are not used in the Iris prototype. This area is left for future research.

Feedback and Errors

One of the frustrations for users of software is the lack of feedback when the system is processing a long task. The user must be kept informed at all stages as to the progress of that task. This may simply involve displaying a message such as:

Processing Account Totals—85% complete…

In more graphical terms, a moving bar that indicates the percentage complete could be displayed. Experiments show that progress indicators are important and enhance the design of the interface (Myers, 1985).

System Response Time (SRT) is the time required for the system to respond to a user initiated event (Rubin 1988). If the length is long (greater than 2 seconds) then appropriate feedback is required. If the SRT is short—too short—the user may feel that the operation was not successful and try to repeat it. In these cases, some form of feedback is still needed. A simple (modeless) text string could be displayed (in a consistent area of the screen) to indicate this.

Another example of feedback is for situations in which functions are available to the user but are not obvious from the interface. That is, the user needs an indication of all possible options that are available. One example is the common use of modifier keys with the mouse button to perform other functions. This can be extremely convenient for experienced users and it should be made clear which options are available at all times.

Ideally it would be better to prevent errors in a software system than to diagnose them at runtime. Unfortunately users seem to push systems to the limit and will always find situations where an error may occur. In the design stage, anticipation of as many error situations as possible is desirable and mechanisms need to be provided to prevent users getting into these situations or display suitable warning messages. Research has found that designing for the prevention of errors is one of the least examined areas for dialogue design (Molich & Nielsen, 1990).

Error messages should be constructive and avoid negative phrasing and obscure codes and should offer the user some constructive help in suggesting a remedy for the situation. Figure 9 shows an example of two error message dialogues with the latter message more informative and including an additional user option to retry the action.

Help

On-line help should be provided at all times and should be easy to find (using a familiar icon or menu option on the screen), and have an extremely simple interface. It should also be easy to switch between on-line help and the work environment. The help system should provide

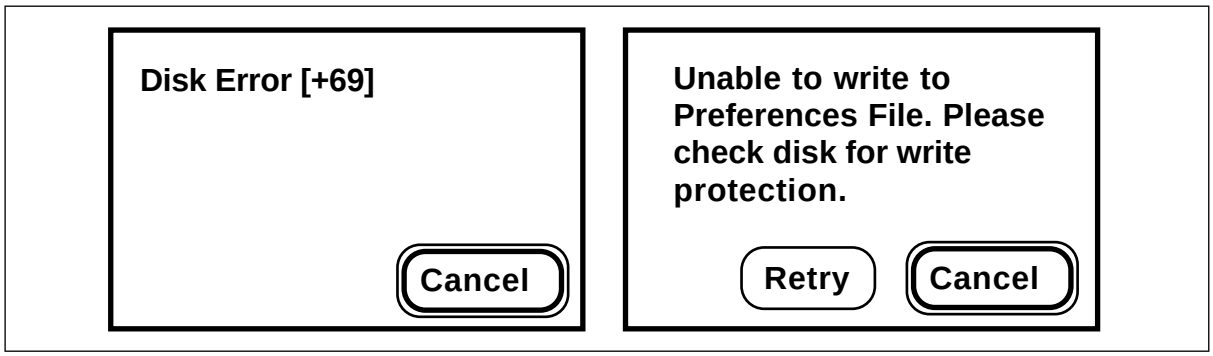


Figure 9: Error Messages Example

explicit guidance in the use of the application and not merely be an on-line copy of the reference manual.

The help system should be context sensitive. For example, if a user is currently drawing a bezier curve in a CAD system, then the help system should display information about curve drawing (with an option to access the full help system as well). Smith (1988) concludes his PhD thesis with the importance that on-line help facilitates to a user interface and the benefits to task performance.

The help system also physically includes the user and reference manuals which, in some cases, may be the most effective means of user guidance. User manuals should provide abundant examples and step-by-step guides to the functions that users are likely to perform. The reference manual should be clearly indexed and provide a quick-reference section for common tasks. Duffy *et al* (1992) review a number of user help environments and provide details on the design requirements for such systems including the evaluation of the level of assistance offered.

In the Iris prototype, the help system was not fully developed. Since the purpose of this research was to design an *intuitive* user interface, the use of help was limited so as to best elicit the most suitable screen designs. The final system, of course, would have a comprehensive help system.

Colour

Colour, if used properly, can be an effective tool to improve the usefulness of GUI design. For example, related items can be grouped together by using similar colour backgrounds and strong contrasting colours can be used to focus attention on critical information. If used incorrectly, colour can impair the appearance and functionality of the system. For example, avoid using spectrally extreme values close together (eg red/green and blue/yellow combinations).

Studies into the physical abilities of the eye have found that the outer edges of the retina are not sensitive to red and green colours and have very few blue-colour receptors (Marcus, 1986). From these two facts, GUI designs should use the following:

- Reds and greens only in the central area of the screen.
- Blues for large areas (screen backgrounds) and not for screen text or thin lines.

The appropriate user-selections of colours is seen to be the biggest problem. With palettes of many hundreds of colours, the user can be bewildered with choices. MacIntyre (1991) proposes a method in which users select colours based on aesthetic and functional properties and not insist on precise colour codes. The user can also specify constraints to modify colours dynamically if the environment changes.

Colour interfaces should be tested on monochrome systems to evaluate the effect of colours under this environment. Some combinations of colours may not be as effective in monochrome and may require the system to be dynamically modified to change the colours displayed. For example, a set of colours may have a fall back set of appropriate grey colours that are used when the software is running on a monochrome workstation.

A conservative set of default colours is used in the Iris prototype (shades of blue and grey). The nature of the OSF/Motif environment also allows for the customisation of colours in X Window System applications. Hence, users may change the background and foreground colours of any part of the screens to satisfy their personal tastes.

Internationalisation

The software industry has an international marketplace, and interface design needs special care when designing a software product that potentially could be used on computing systems in any country. Consideration of the internationalisation of the interface must be included for cultural differences, language, and local conventions (Del Galdo, 1990).

Provisions should be made in the software (with a customisation utility for the user or distributor) to handle differences in a number of areas. These include:

- Numeric formats, including different date and time standards.
- Icons and symbols which may require testing of usability across different cultures.
- Fixed screen text which, after language translation, may expand and require increased screen real estate.
- Menu accelerators that follow a mnemonic structure may be difficult to translate into other languages as well as different country keyboard layouts.
- Support for large character sets (such as the 4,000 Kanji characters) and different collating sequences of characters.

An interesting example of cultural differences with icons and colour involves the use of a mailbox icon for representing an electronic mail system. The problem arising is that a mailbox is not consistent throughout all countries as a general representation of mail. There are also differences for the colour of mailboxes; blue in USA, red in England, and yellow in Greece. In this case a more appropriate iconic metaphor may be an envelope of some neutral colour (see Figure 10).

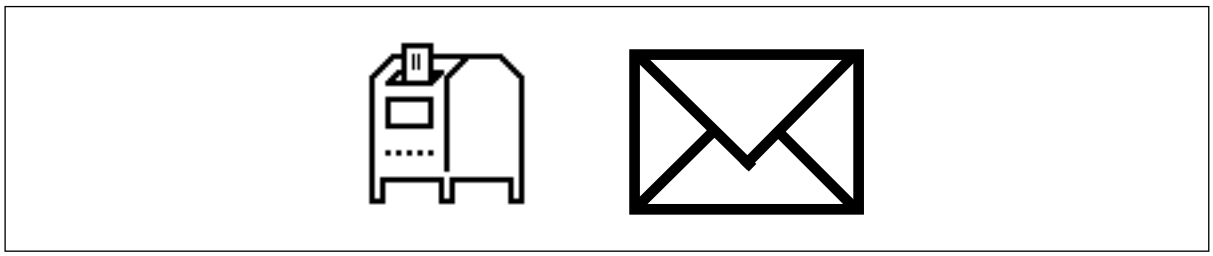


Figure 10: Electronic Mail Icon Examples

Fortunately most of the user interface toolkits provided today have inbuilt support for some aspects on internationalisation. Usually all user interface translatable items can be set up as country specific resources that can be easily modified and distributed with the software system. For example, all text messages that are displayed should be indexed and called appropriately from a country specific resource file—never 'hardwired' into the application code. Weir

& Qing (1993) describe such a system in which the screen instructions and help files can be easily switched between English and Chinese for different users.

Another solution is to treat the translated interface as a *new* interface (Nielsen, 1990a) and to re-evaluate the interface with the new cultural audience. See '2.8 Evaluation and Usability' on page 42. This will involve more effort with better results.

Fortunately, underlying operating environments, such as OSF/Motif and Macintosh, have intrinsic support for internationalisation issues. Such applications, if developed to specification, can automatically use known conversions for some of the above features.

Examples

MicroStation Mac[3] is a Computer Aided Design (CAD) program for the Macintosh which, at first glance, seems bewildering. The system relies heavily on the concept of modeless iconic 'tool' palettes. In fact, over 285 different icons are used throughout the software, a sample of which can be seen in Figure 11. It goes one step further and uses hierarchical palettes. These are sub-palettes that can be 'torn-off' and placed on the workstation screen. Some of the icons even have 'pop-down' information tables that allows for extra information to be supplied for some of the functions (see the 'Line Strings' palette in Figure 11).

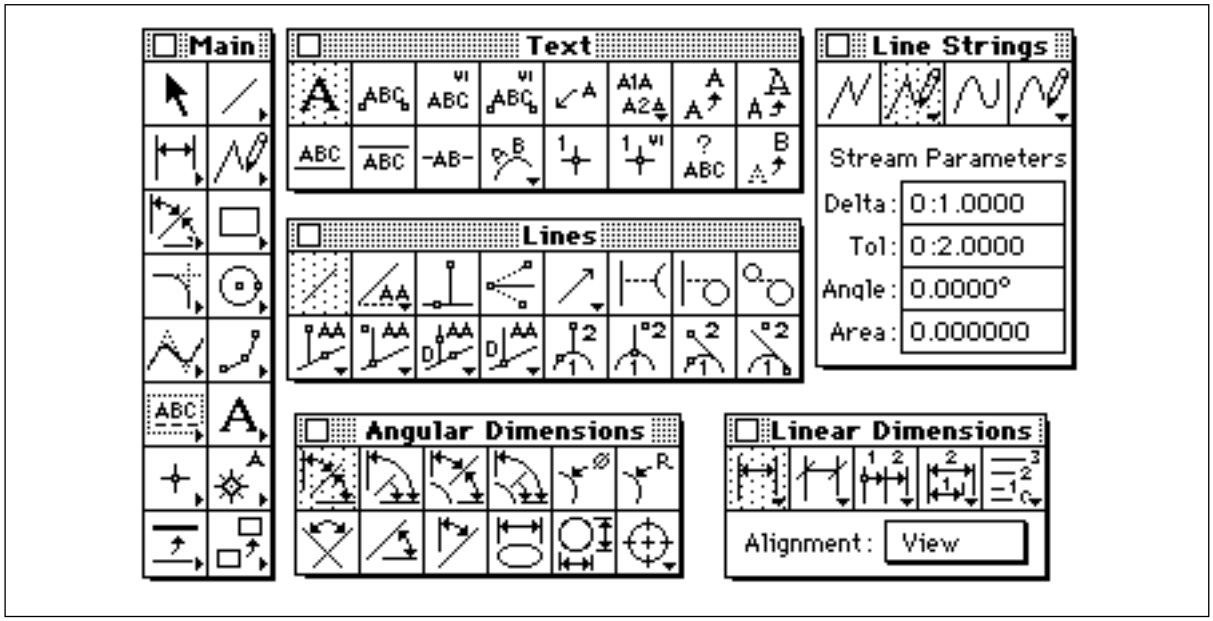


Figure 11: MicroStation Mac Icons

The designers of MicroStation Mac seem to have been quite aware of the possibility that users may need assistance as the functionality of each icon and added a Command Window that serves as a feedback tool (see bottom of Figure 12). Whenever an icon is selected, a description of the active command is displayed in the Command Window as well as supplementary information as to what is required next (eg 'Enter first point'). The current environment settings, feedback and any error messages are also displayed in the Command Window. Another helpful tool provided by the designers is the Command Browser (see top of Figure 12). If the user is unsure of a command syntax, the Command Browser can be popped up (by typing '?' into

3. MicroStation Mac is a registered trademark of Intergraph Corporation

the Command Window) and the 'point-and-click' approach can be used to construct the correct syntax for the desired command.

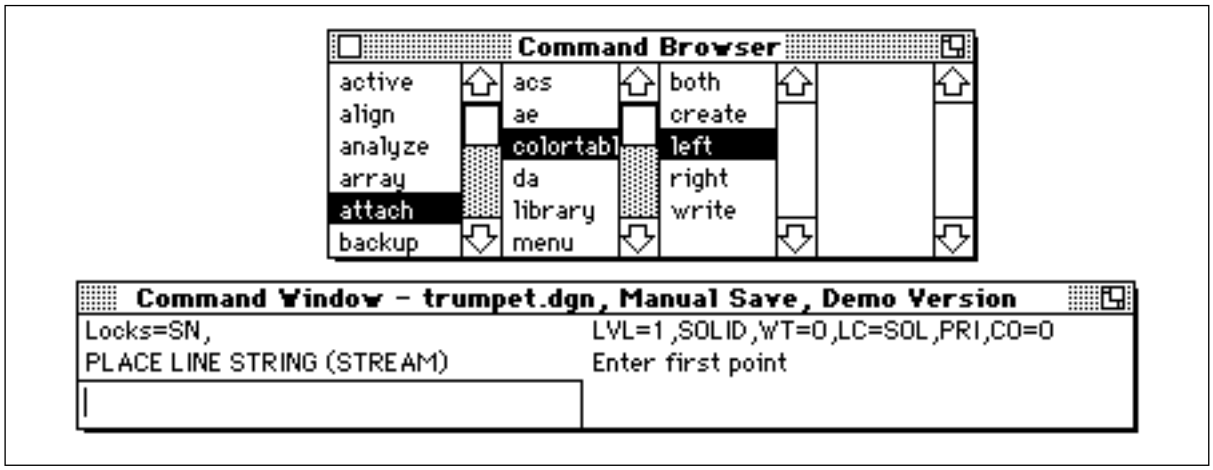


Figure 12: MicroStation Mac Command Browser and Window

Overall, MicroStation Mac provides excellent feedback and assistance to the users of a very complicated system. Even Macintosh-literate but CAD-illiterate users could master the basics of the system within a short period. The manuals (Intergraph, 1990) and on-line help are well organised and an electronic hypertext version of the Reference Manual is also provided.

As a negative example of user interface design, consider the screen images from the publishing system FrameMaker[4] (Macintosh version 3.0.1i) in Figure 13 and Figure 13. FrameMaker has a catalogue of formats for characters, paragraphs, and tables. Figure 13 shows the Paragraph Format dialogue (with the Properties option menu raised) and the Paragraph Catalog palette. Notice that a 'Delete…' button is provided (to remove a paragraph format) on the bottom of the Paragraph Catalog palette. A similar option is also provided on the Character Format palette.

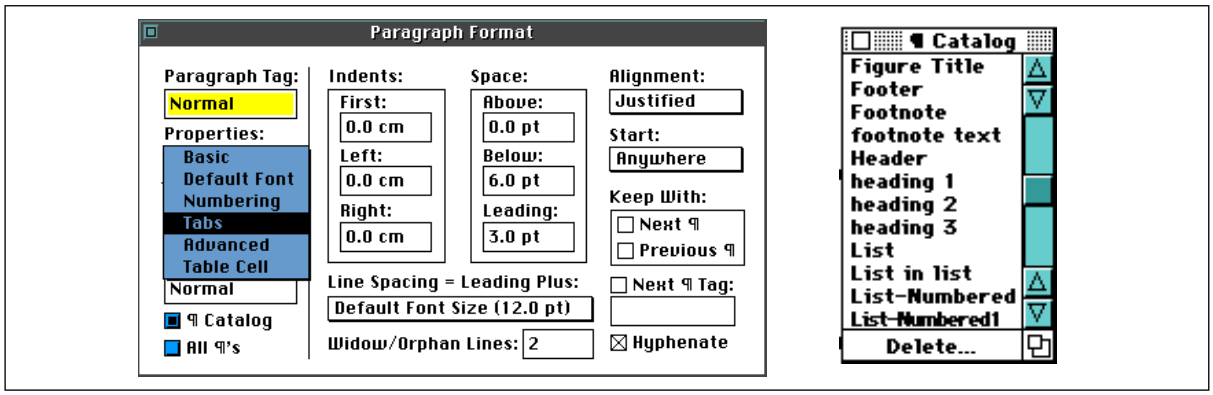


Figure 13: FrameMaker Paragraph Format and Catalog Windows

Figure 13 shows the Table Format dialogue (with the Properties option menu raised) which is similar to the Paragraph Format dialogue. Notice that there is also a 'Delete from Catalog…' option available in the Properties option menu.

The problem with these dialogues is the inconsistent method used to delete paragraph and table formats. Both have similar use and functionality in the software but the table formats em-

4. FrameMaker is a trademark of Frame Technology Corporation

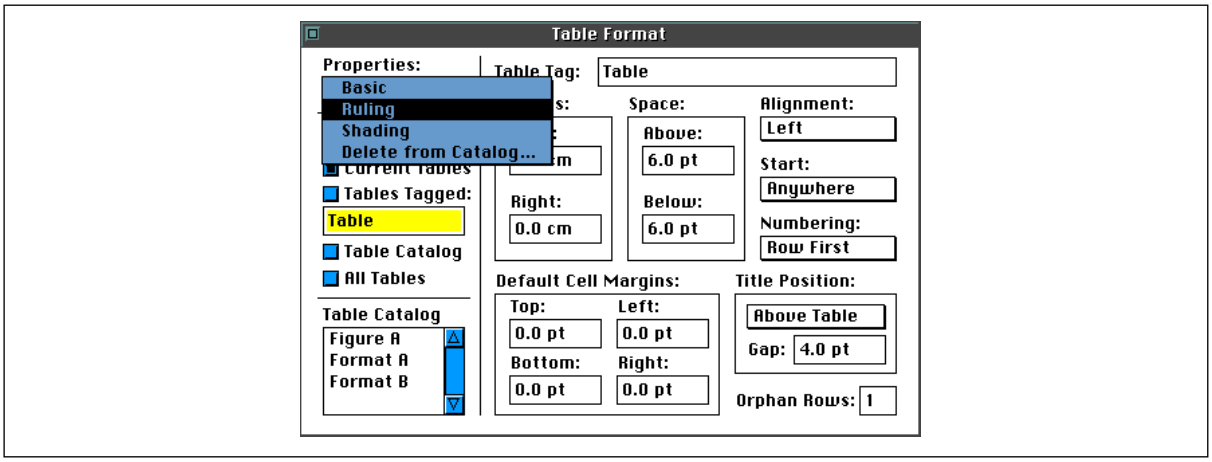


Figure 14: FrameMaker Table Format Window

ploy a different method for deletion. Even worse is the fact that the deletion of a table format now has been classified as a 'property' of a table format.

Solutions to this particular design problem would be to:

1 provide a Table Format palette window (the most consistent solution), or
2 provide a separate button to delete Table Formats.

2.5.5 Consistency

As was shown in the previous FrameMaker example, consistency must be a major aim for user interface designers. Consistency for the user interface can be thought of in terms of (Koritzinsky, 1989):

- Predictability—the user can anticipate what the system will do
- Dependability—the system will fulfils the user's expectations
- Transferability—the usage is similar across applications

It is obvious that the user would benefit most from consistent user interfaces mainly through transferring the same skills across heterogeneous systems. An organisation can also benefit from employing a consistent user interface standard. This includes lower training costs for employees, reduced costs, and higher quality for systems development over the long term (Nielsen, 1989a).

Consistency can be achieved in a number of ways:

- Appearance—consistent spatial and visual characteristics
- Metaphor—consistent and *appropriate* metaphors used
- Behaviour—consistent behaviour by similar interface components
- Dialogue—consistent terminology

Tools that can aid in the development of consistent interfaces include company style guides, user interface standards, prototyping, and User Interface Management Systems (see '2.6 User Interface Management Systems' on page 30).

Some researchers have argued against user interface consistency. Grudin (1989) argues that consistency is an unreliable guide and that designers should concentrate primarily on the user's work environments. This discussion has reflected the dissatisfaction with the common

sense meaning of consistency and the possible reduced efficiency of consistent interfaces (Wiecha, 1992; Grudin, 1992).

Another possible problem with consistent interfaces across applications is when interfaces are copied between applications for the sake of consistency. The highly competitive market place has led some companies to provide 'excessively similar' interfaces for the same class of application. This has resulted in many legal cases and the issue of user interface copyright (Stern, 1989).

As is clearly shown, the issue of consistency requires a balanced trade-off between ethical responsibilities and user acceptance. Achieving a consistent user interface is a formidable goal for any designer and is well accepted and a laudable one. This research has adopted user interface consistency within and across the Iris prototype as a high priority. If used sensibly, consistency will increase the usability and functionality of software.

2.5.6 Direct Manipulation

Direct manipulation interfaces are interfaces in which the user has direct control over the objects on the screen. The user's intentions can be translated into direct actions on these objects usually reflecting some real world metaphor. The result of such actions are immediately obvious and visual feedback shows the progress of such actions. This is the area in which the GUI has provided the most benefit with high resolution graphic screens and various input devices.

Direct manipulation interface styles increase the user capabilities whilst reducing the cognitive overheads. For example, to delete a file on a Unix system a user must issue the command '`rm`' followed by the file name. The same command on a DOS system is '`del`'. On a Macintosh computer, the user directly engages with the file system and drags the selected file icon to the rubbish can icon. Research has shown, in this case, that the direct manipulation method is more productive (Israelski *et al*, 1989). Conflicting research (Benbasat & Todd, 1993) shows that direct manipulation interfaces may initially be faster than menu-based interfaces but the benefits may diminish as the learning period increases.

Hutchins *et al* (1986) specify that to produce the feeling of direct engagement a system must provide the following:

- minimal gulfs of execution and evaluation,
- interreferential manipulation (which allows an input expression to make use of a previous output expression),
- a responsive system with no delays between execution and results, unless appropriate for that knowledge domain, and
- an unobtrusive interface that does not interfere with the directness of engagement.

In some interfaces it has been shown (Ankrah *et al*, 1990) that direct manipulation techniques can still be used that do not necessarily reflect any real world metaphor. Hartson *et al* (1990) have developed a language called User Action Notation (UAN) for behavioural representations of direct manipulation interfaces. UAN provides a precise description of the user actions

and system feedback. An example of UAN is shown in Table 2 and represents a user selecting a file icon with the mouse, moving it to another location, and releasing the mouse button.

Table 2: UAN Example

TASK: move a file icon	
User Actions	Interface Feedback
~[file_icon] Mv	file_icon!
~[x,y]* ~[x′,y′]	outline of file_icon follows cursor
M^	display file_icon at x′,y′

Williamson & Shneiderman (1992) describe the use of direct manipulation techniques in formulating queries for a database application. These 'dynamic' queries allow the user to adjust graphical objects and view the results immediately. This method has shown many advantages for the user over the traditional methods of query formulation.

2.5.7 Intelligent Interfaces

An intelligent user interface is one in which the application uses a knowledge base to *change* the interface depending on the user's experience, preferences, and the context of the current dialogue. This change may involve an addition of user interface options to increase functionality or a reduction, to lessen the burden of excessively sympathetic interfaces. Such interfaces usually involve artificial intelligence techniques to be successful. Shneiderman (1989) feels that intelligent interfaces can be misleading as sudden system initiated changes would disorientate the user and undermine the user's sense of control and predictability.

A more popular concept of interface 'agents' is beginning to emerge as a preferred option to incorporate intelligence into user interfaces. Studies have shown (Wahlster, 1989) that novice users of systems with cooperating intelligent agents can complete tasks with encouraging results. The agent acts on behalf of the user in the carrying out of a well defined task and hence the user does not feel to have lost control. Early attempts at agents would simply look for repetition of user actions. Once detected, they would offer the user to perform the same task automatically. Rodden *et al* (1992) used autonomous agents with inherent decision-making capabilities to interact with the human user or other agents as an implementation strategy for a distributed object-oriented environment.

Naffah *et al* (1989) believe that the model for user interfaces needs to change if intelligent interfaces are to be used. They believe in a fully object-oriented system that has a three layer architecture:

1 Nodes (application objects or their views)
2 Gadgets (library of graphical tools)
3 Drawables (output) and Editors (input)

Systems they have produced using this model on the X Window System have proved successful.

A special type of intelligent interface is the adaptable interface. The adaptable interface supports a number of different dialogue modes that the user can switch between easily and at any time (even in the middle of a command sequence). Kantorowitz & Sudarsky (1989) used an

adaptable interface to test their GUIDE system in which users can interact with a menu based or a command based dialogue. They found that the freedom for the user to adapt dialogue modes was useful at all levels of user experience. Benyon & Murray (1988) describe an attempt at an *automatic* adaptive interface to their MONITOR system. In this case, a series of scripts described rules that changed the user model and interface based on certain knowledge of the user and the current state of the system. The bottleneck they found was in eliciting users cognitive traits to base the interface models on. Danielsen *et al* (1991) discuss an adaptive user interface to computer-mediated communication services called Ratatosk. Ratatosk uses a plan-recognition algorithm which leads to a decrease in response time as (in most cases) the system will know what the users want to do before they request it.

2.6 User Interface Management Systems

One of the key motivations of using a User Interface Management System (UIMS) is the separation of the interface from the application code. Using a Computer-Aided Software Engineering (CASE) tool such as a UIMS has allowed this separation to be enforced and encouraged UIMS deployment by developers. Traditional methods of separating the interface and application into source code files has not always been successful. A UIMS manages the interface as well as the methods to communicate with the application code. The UIMS is perhaps the most important software development tool for the design and implementation of the user interface (Rhyne *et al*, 1987).

A UIMS is defined by Betts *et al* (1987) as:

> *'…a tool designed to encourage interdisciplinary cooperation in the rapid development, tailoring and management (control) of the interaction in an application domain across varying devices, interaction techniques, and user interface styles.'*

Motivations for a UIMS range from the complexity of designing powerful user-enabling GUI's and the amount of code required to implement such interfaces. Results have shown (Myers & Rosson, 1992) that some 48% of the code being developed is specific to the user interface. A UIMS requires the development of models and algorithms for the design of user interfaces. Olsen (1992) discusses a number of these key algorithms used to automatically manage user interfaces within the framework of a general UIMS architecture.

Betts *et al* (1987) also outline the benefits of a UIMS as:

- Consistency
- Support for a range of users
- Support for error handling and recovery
- Support for tailorability and extensibility

Many UIMS have been developed using both early user interface models and new models that attempt to capture new development metaphors. Hartson & Hix (1989b) developed the *star lifecycle* to support UIMS activities and bridge the interface development gap. Unlike the conventional 'waterfall' model, this model supports alternative approaches to development that are reflected in UIMS that support some level of interface evaluation. Lantz *et al* (1987) also postulated a high level model for UIMS incorporating a dialogue manager, workstation manger, workstation agent, and application that provided a framework for implementation strategies whilst emulating the OSI Reference Model. Lantz (1987) further refined this model to structure user interface software as multiple cooperating processes. This approach would be

more flexible than the current UIMS approach to generate monolithic amounts of code that have a high redundancy and performance overheads.

Myers (1988) lists the following advantages of UIMS:

- The design can be rapidly prototyped.
- The quality of the overall interface increases since changes can easily be incorporated.
- A UIMS provides a more consistent interface across and within applications.
- The interface specifications can be represented, validated, and evaluated more easily.
- A UIMS can support a number of roles for individuals on the development team.
- Non-programmers can easily develop and modify interfaces.
- A UIMS supports the separation of management of dialogues and application.
- The reliability of interfaces should be higher due to automatic code generation.
- Some cost savings are possible due to reduced development time.

These advantages do not come automatically and still require proper management of the development team. For example, human factors analysis still has to be performed to achieve consistency in the interface even whilst using a UIMS (Bennett, 1987).

Some problems and limitations of UIMS include the following:

- A UIMS may require the developer to learn a new special purpose interface language.
- A UIMS may limit the types of interactions available on the system. For example; gestural interfaces (Rhyne, 1987).
- Some graphical UIMS may not support multiple interactions.
- A UIMS lacks ways of presenting application data that may require display update facilities (Olsen, 1987).
- The application programmers may lose some control over program structure and there is little support for underlying data structures (Kasik *et al*, 1989).

UIMS can be broadly categorised into two types:

1 Command—in which the interface is specified in some lexical form from a predefined interface language syntax.
2 Graphical—in which the interface is specified by direct manipulation of interface objects.

Within these two groups there are different levels for support of the tailorability of the dialogue as well as the style and sequence of the dialogue. Some UIMS also support the testing of the interface via the generation of code and/or 'on-line' running of the application in a simulation mode. The latter method gives the developer immediate feedback as to the dialogue interaction and sequencing. The more successful UIMS also provides semantic feedback. In this case, the UIMS emulates the interface with more meaningful parameter values from inherent knowledge of the interaction styles and interface component relationships (Dance *et al*, 1987). Carey & Crentsil (1992) describe how user interface design knowledge can be presented to designers as part of the interface widgets used in the NeXT Interface Builder UIMS. The designers had access to an aid that supplied information on the viable use of each widget as well as other options and a ranking in terms of the widget's usability.

Hudson (1987) raised the issue that since direct manipulation offers significant advantages to novice users, graphical UIMS should minimise and de-emphasise reliance on rigid syntax.

This would make UIMS more accessible and flexible than other design tools. Myers (1987) also noted that the textual based UIMS were hard to use and that the direct manipulation UIMS are more functional.

2.6.1 UIMS Examples

UIMS can be classified into research-based (those usually developed at universities and research institutions) and commercial products.

Research UIMS

There has been a plethora of UIMS developed in this category, mainly to satisfy specific UIMS research needs. The GRINS system (Olsen *et al*, 1985) is an early example of a command-based UIMS that provides linkage between the logical device interface and the graphical representation of virtual devices. GWUIMS (Sibert *et al*, 1986) is an early example of a graphical UIMS that was designed using the object-oriented programming paradigm and consists of a variety of objects classes representing different levels of abstractions. A message passing system is also used to emulate the interface semantics.

OSU (Lewis *et al*, 1989) is an example of a fully graphical UIMS that supports all interface actions via direct-manipulation. In some cases, graphical aids were developed for command based UIMS. OPUS is a graphical user interface editor that generated the appropriate (textual) interface notation to be used in the command based Penguims UIMS (Hudson & Mohamed, 1990).

MoDE (Shan, 1990) is an example of a UIMS that provides support for an extensible interface library and connection to the underlying application. The user can easily add new objects to the interface library and use them in the interface. MoDE allows the user to attach a 'semantic object' to each interface component. The semantic objects are layered between the application and the interface and have knowledge of both. The Chimera UIMS (Wood & Gray, 1992) also attempts to solve this problem by using interprocess communications between the user interface and application. In this case a series of Unix pipes are used in the communication of the interface semantics.

Perdiot (Myers, 1990b) is an example of a UIMS that is based on the 'program by example' principle. The designer draws a picture of the interface and uses direct manipulation to demonstrate how the interface should operate. Perdiot then generates parameterised procedures including any user defined constraints.

The UofA* UIMS (Singh & Green, 1991) addresses the problem of the initial design of the user interface. The UofA* approach is to produce the initial design automatically (based on a series of default values) and then enable the designer to improve the appearance through the usual refinement process. Encarnacao *et al* (1991) describe the use of artificial intelligence in UIMS and the role a knowledge base can provide for both the designer and interface being developed.

Barker *et al* (1990) describe the eXCeS UIMS and its evolution into a sophisticated tool for modelling dynamic systems with links to external numerical simulations. Originally eXCeS used the GKS graphics standard but now supports both OpenLook and Motif interface widgets. Both were supported because it was unclear at the time as to which interface toolkit would become the de facto standard. TAE Plus (Szczur & Sheppard, 1993) provides rapid prototyping,

evaluation, and implementation of user interfaces using the X Window System and is tailored for support of real-time control and processing applications.

The research based UIMS have highlighted and attempted to solve some of the key issues in UIMS. Some of these solutions are beginning to appear in commercial UIMS. It is interesting to note that one of the key benefits of UIMS is the separation of interface from application which is one of the areas that needs the most attention. Absolute separation will hide the interface semantics and led to subsequent implementation problems. An appropriate mix is required. Treadway (1989) sees this separation as the single most critical underlying theme in UIMS and one that open computing technologies must embrace. Neilson & Weir (1987) provide a summary of four UIMS that embrace the philosophy of dialogue separation but warn of the practicality of a full separation of interface and application.

Commercial UIMS

The market for commercial UIMS is mainly focused on direct manipulation graphical systems. It is a very competitive market with many new products being released annually which usually reflects the changes in the available user interface toolkits and sophisticated needs of interface developers. Prime (1990) and Sastry (1992) provide comprehensive summaries of available commercial UIMS. Kuhme & Schneider-Hufschmidt (1992) describe the SX/Tools UIMS for adaptable multimedia user interfaces. Borras *et al* (1992) describe O_2Look, a UIMS for the generation of database application interfaces. Foody (1989) describe the UIMX UIMS that will automatically apply corporate style guidelines to interfaces developed with it.

Figure 15 shows a screen image of the XBUILD[5] UIMS. On the left of the screen is a list of interface components (in this case, the OSF/Motif widget set) which the user can drag onto the large work area on the right. In this example, the user has put together a dialogue with a number of buttons and text fields. XBUILD allows easy modifications on the attributes for each interface component and on the bottom right of the screen is the 'test mode' button. XBUILD generates standard C code from the interface and the widget 'callback' stubs for the user to develop the application code. XBUILD was one of the first UIMS for the OSF/Motif widget set but has been found to be an unreliable system (ie unexpected crashes) and has since been discontinued by its manufacturer.

Figure 16 shows a screen image of the VUIT[6] UIMS. On the left of the screen is the list of OSF/Motif widgets. In this case, the widgets are presented in a hierarchical fashion (clicking on the arrow will show all the subordinate widgets) and graphical icons are used for each widget. On the right of the screen is the work area in which widgets are dragged and placed appropriately. VUIT also offers full customisability of all interface components and a simulation mode. VUIT generates User Interface Language (UIL) code and 'callback' stubs for the application code. The UIL is a standard language for the specification of interfaces.

Both XBUILD and VUIT heavily rely on the designers knowledge of the OSF/Motif widget set. This is the same case for other UIMS with different user interface components. For example, in creating a pulldown menu, the designer must specify each sub-component of the menu in the exact order, otherwise the UIMS will report an error. There is no support for high level interface abstractions or for interface semantics.

---

5. XBUILD is a registered trademark of Siemens Nixdorf Information Systems
6. VUIT is a registered trademark of Digital Equipment Corporation

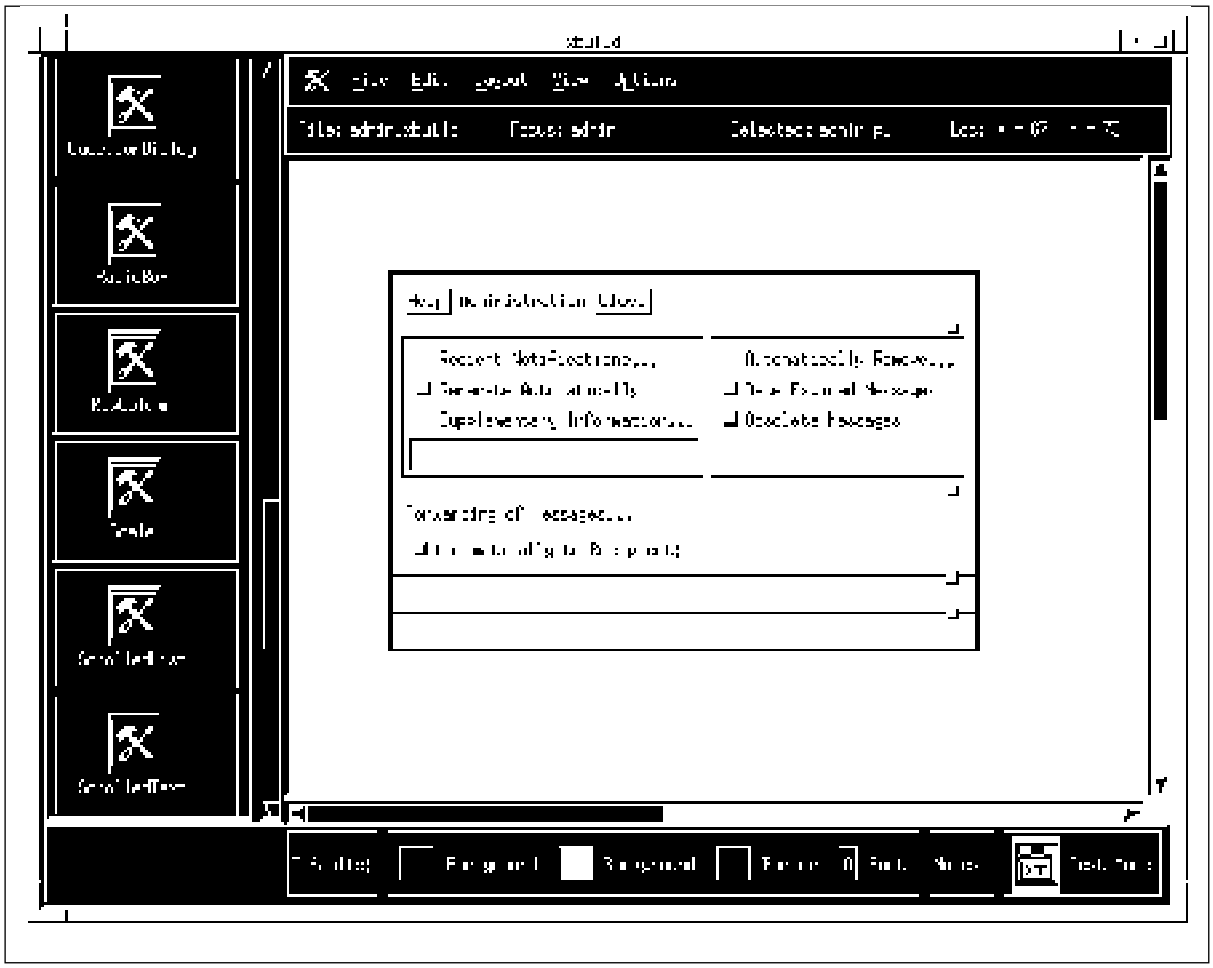


Figure 15: XBUILD UIMS

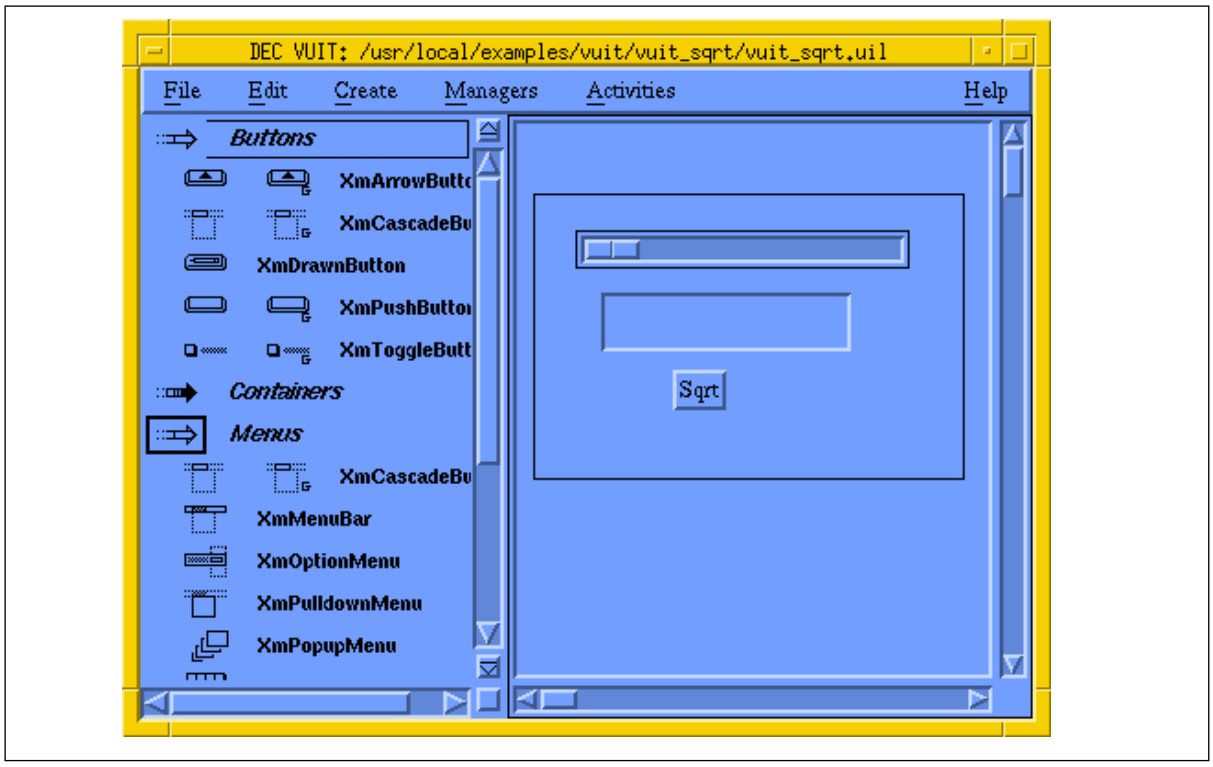


Figure 16: VUIT UIMS

In general, commercial UIMS do provide many of the benefits previously mentioned but are not as easy to use without the inherit knowledge of the interface toolkit. The trend is changing and such requirements are being met in new software releases.

2.6.2 Prototyping

A major benefit from using a UIMS is in prototyping the interface. Prototypes may be either functional or user interface mock-ups. A functional prototype will attempt to mimic all the functional requirements of the system. A user interface mock-up will focus on the interface and utilise 'smoke and mirrors' to simulate the functionality. Since the UIMS usually allows an interface to be developed more quickly than traditional methods, the interface can be tested with end-users. Any changes can then easily be made since it does not affect the underlying application code.

This iterative process involves developing prototypes of the software system, demonstrating to the end-user, and incorporating any feedback from the users into the next version of the prototype. Figure 17 shows a comparison of the traditional development process and one with prototyping included.

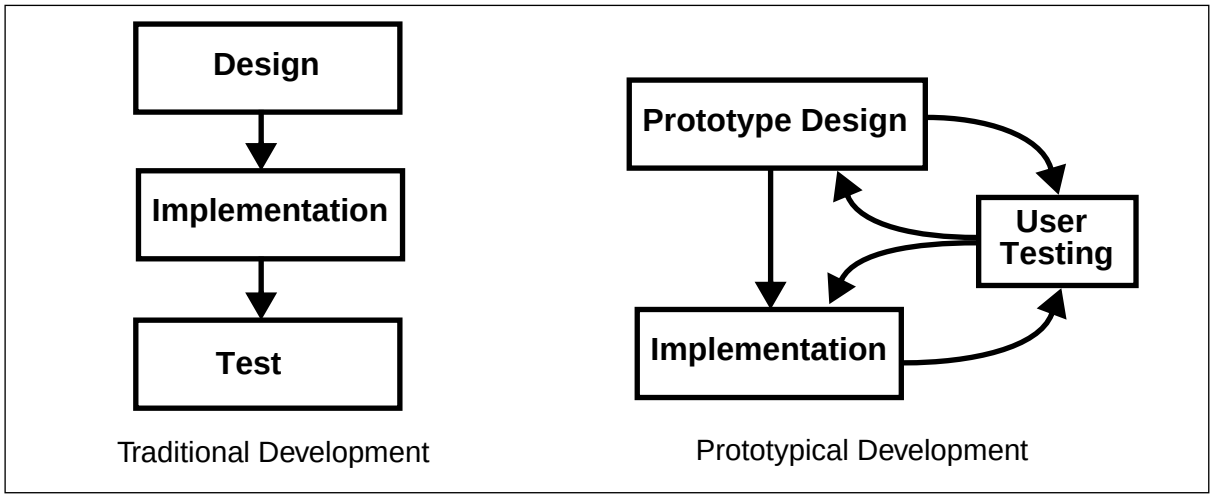


Figure 17: Traditional and Prototypical Development

The benefits of prototyping (Wasserman & Shewmake, 1990):

- enable the *user* to evaluate the interface and suggest changes.
- enable the developer to evaluate user performance.
- enable experimentation with alternate interface styles.
- give the user an immediate sense of the proposed system and encourage them to think more carefully about their needs.

These benefits allow the designers to be confident of the interface and increases the success of the project. Hence, prototyping should be part of the systems development cycle. Mock-up prototyping improves the quality of the user interface as designers can evaluate alternate interfaces at an early stage in the development (Benimoff & Whitten, 1989). Functional prototyping can also identify any enhancements and additional functions of the system.

Prototyping does have some potential pitfalls. One of the most serious is when the development team view the prototype as the finished product (Hartson & Smith, 1991). There is a temptation to use the prototype as the final system if time and other resources are diminish-

ing. This usually leads to disappointment as prototypes are usually slower and have inherent disadvantages over the 'real' system.

Even though a UIMS can be used as a prototyping tool, there exists specific functional and user interface mock-up prototyping software for the rapid development of interfaces (and not the generation of application code). These include:

- HyperCard[7] for the Apple Macintosh
- Toolbook[8] for DOS-compatible machines
- MetaCard[9] for Unix/X Windows

Figure 18 shows a screen image of HyperCard. It has extensive tools to rapidly develop an interface as well as a scripting language to emulate the final system. HyperCard has been a very successful prototyping tool, so much so that it is quite often used by users to develop highly customized applications.

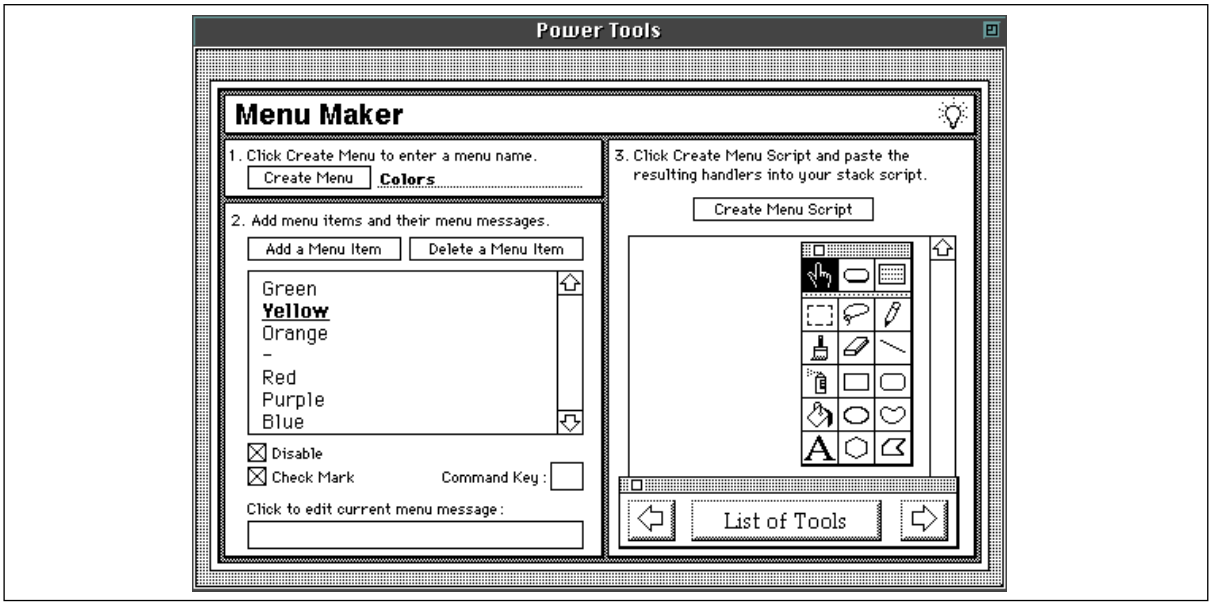


Figure 18: HyperCard

2.7 Guidelines and Standards

There is little question as to the benefits of user interface design guidelines and standards. The development of corporate user interface guidelines and standards appears to be both technologically and financially justified (Rosenberg, 1989) due to the increased productivity for both user and manufacturer. Guidelines and standards exist in many forms and are applicable to a wide area of the user interface. It is usually a question of discovering the guidelines or standards and extracting the relevant ones to the system being designed.

One of the potential problems with guidelines and standards is the possibility that developers may use the recommendations as an excuse not to test their interface (Potosnak, Sept 1988). The usual testing procedures still apply to the *entire* interface even if close attention to individual guidelines and standards have been followed. Another major disadvantage of guidelines

7. HyperCard is a registered trademark of Apple Computer
8. Toolbook is a registered trademark of Asymetrix Corporation
9. MetaCard is a registered trademark of MetaCard Corporation

and standards is the possibility for a mismatch between the user interface issue and the relevance of the recommendation. Since most guidelines and standards will require the designer to interpret the recommendation and apply it to the system under review, the potential for mismatch is possible. Nevertheless, guidelines and standards are a rich source for recommendations and, if used correctly, can be effectively used for the design of user interfaces.

There are generally two categories of guidelines and standards:

1 those that are specific to particular interface environments, and
2 those that are general and apply to all interface environments.

Standards that are set by national and international bodies usually fall into the latter category of general guidelines.

2.7.1 Standards

It is often (humorously) quoted in the computer industry that:

'…the good thing about standards is that there are so many to choose from.'

Unfortunately this does apply to the user interface sector as there is an abundance of standards and an equal number of draft standards. The International Standards Organisation (ISO) alone has developed approximately 70 standards with the same number of draft standards still to be released (Pangalos, June 1992). Each standard covers a specific area of the user interface. Figure 19 shows the breakdown of ISO standards in each area of user interface design (Pangalos, 1992). Other national and international standards bodies have similar numbers of existing and draft standards. Standards have also been developed by the US Department of Defense (1989) that outline the design criteria for military systems.

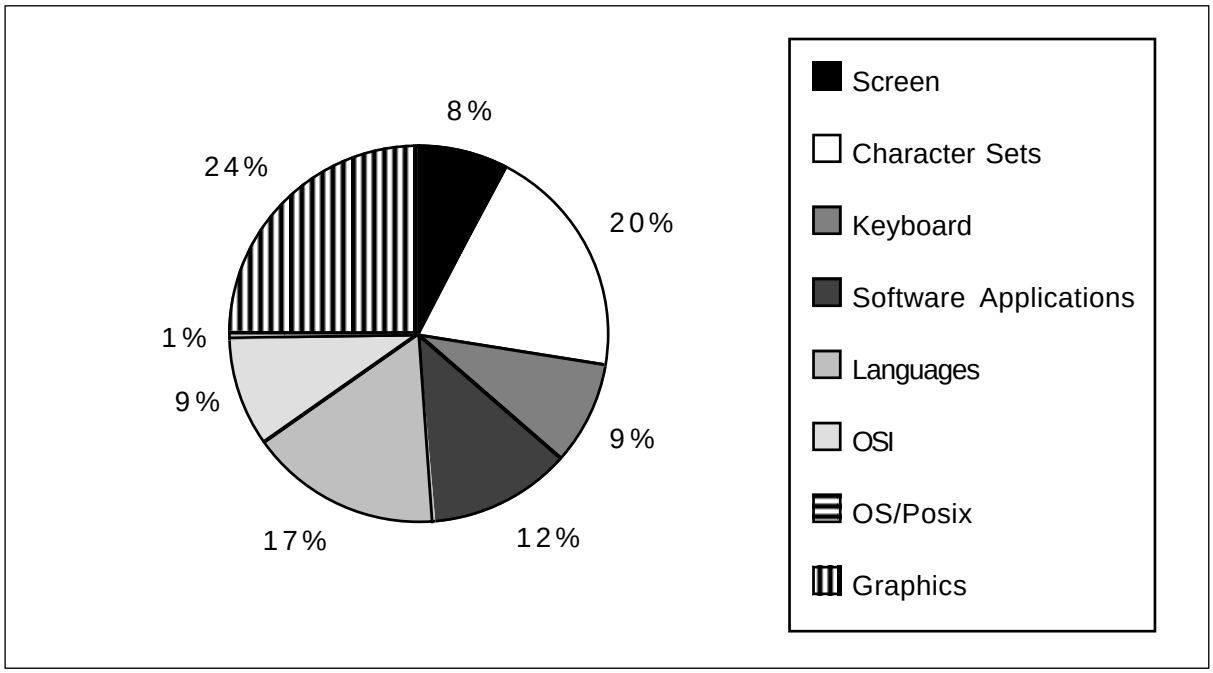


Figure 19: ISO Standards breakdown

These standards have not been widely used to date, but with the movement towards the legal requirements to follow such standards, it is only a matter of time when user interfaces will conform. Bevan (1991b) outlines the ISO 9241 draft international standard and its impact and relevance on the European Directive for conformance by the end of 1992. Stewart (1991) also

reviews the ISO 9241 standard and outlines a twelve point strategy in applying and reviewing the recommendations as a platform for making usability an integral part of the system design process.

2.7.2 Guidelines

There are a number of steps in applying guidelines that should be followed (Potosnak, Jan 1988):

1 Select guidelines that apply to the system being designed (not all would be relevant).
2 Determine the ranking of important guidelines.
3 Translate each guideline into rules that can be directly applied (by programmers).
4 Plan for any exceptions to any guideline.
5 Evaluate the design.

There are numerous sets of guidelines that cover a wide range of computer environments and interfaces. Almost all need to be subjected to the above checklist. Others, that are specific to particular interface toolkits or areas (eg colours) can be used in a more generalised way.

Smith and Mosier Guidelines

Smith & Mosier (1986) produced the most comprehensive set of guidelines for computer systems interfaces which included 944 individual guidelines detailing recommendations on the design of the user interface. The report by Smith and Mosier (SAM[10]) recommended guidelines for user interface design in six functional areas as shown in Table 3.

Table 3: SAM Guideline Areas

Section	Area	Number
1	Data Entry	199
2	Data Display	298
3	Sequence Control	184
4	User Guidance	110
5	Data Transmission	83
6	Data Protection	70

Each guideline is structured with up to six fields:

1 Statement
2 Comment
3 Exception
4 Example
5 See Also
6 References

The Statement field briefly outlines what the guideline covers and the Comment field presents more information on applying this guideline to interface design. The Exception field suggests

[10] The 'A' in SAM refers to Arlene Aucella, who worked on the guidelines with Smith before Mosier.

any restrictions on the use of this guideline. The Example field shows an example of the guideline, usually contrasting good and bad interfaces. The See Also field mentions related guidelines in the report and the References field lists external documents that the guideline has been extrapolated from. An example of one of SAM's guidelines is show in Figure 20.

Section 2: **Data Display**
Sub-Section 0: **General**
Number 3: **Data Displayed in Usable Form**

Statement:

Display data to users in directly usable form; do not make users convert displayed data.

Comment:

Do not require a user to transpose, compute, interpolate, or translate displayed data into other units, or refer to documentation to determine the meaning of displayed data.

Example:

If altitude might be required in either meters or feet, then display both values. This recommendation applies to error messages and other forms of user guidance as well as to data displays.

(Probably adequate)

```
Character in NAME entry cannot be recognized.
```

(Too cryptic)

```
Error 459 in column 64.
```

See Also:

4.4/1 Guidance Information Always Available

Reference:

Brown et al 1983 § 3.3
Engel Granda 1975 § 3.3.4
MIL-STD-1472C 1983 § 5.15.3.1.3

Figure 20: Example SAM Guideline

With such a large document, alternative computer-based approaches have been utilised to present the guidelines to designers in a more manageable form. Using hypertext techniques for cross-references, these computer-based interfaces have been instrumental in providing large databases of guidelines to interface designers. The applications include:

- NaviText[11] SAM (Perlman & Moorhead, 1988; Perlman, 1987)
- DRUID (Fox, 1991)
- BRUIT∆SAM (Iannella, 1992a)

An example of a SAM guideline from BRUIT∆SAM is shown in Figure 21.

The SAM guidelines have been used successfully for the development of many systems (Toleman & Welsh, 1991). The guidelines do not cover graphical interfaces, but provide sufficient

[11.] NaviText is a trademark of Northern Lights Software Corporation

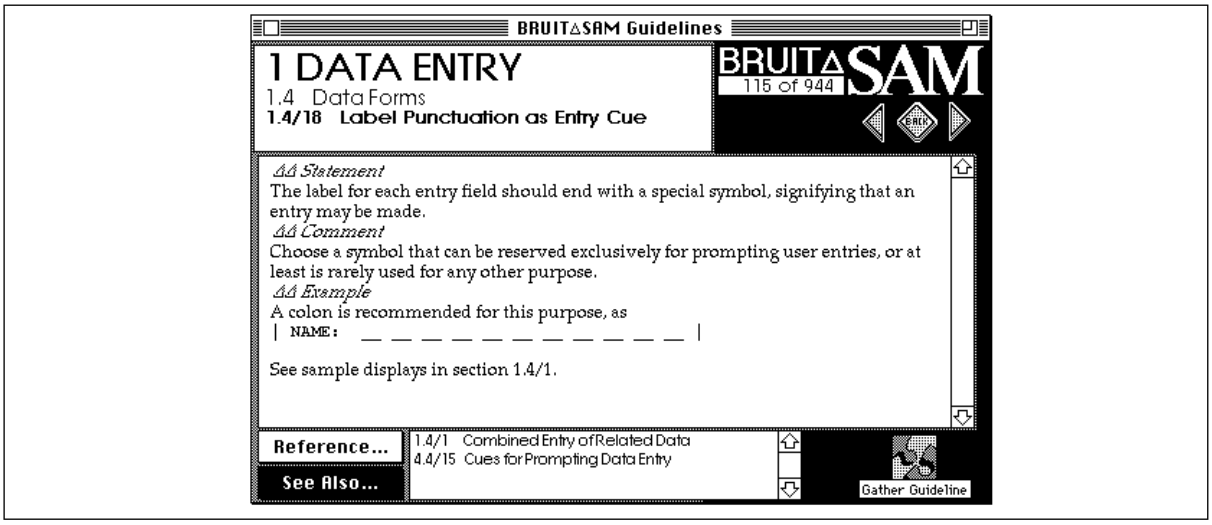


Figure 21: BRUITΔSAM Guideline

guidance that applicability to graphical interfaces can be implied. The SAM guidelines are used extensively throughout the design of the Iris prototype.

Common User Access Guidelines

Common User Access (CUA) establishes a degree of standardisation across IBM's range of Systems Application Architecture (SAA) computer environments. It provides SAA users with a consistent interface across graphical workstations and text-only terminals. Some of the CUA design principles that are unique to SAA include the following (Berry, 1988):

- CUA takes advantage of graphical workstations were possible. The CUA is not designed on the lowest common denominator solution.
- CUA efficiently uses the keyboard interface for office applications. The user should not be required to use a mouse for interaction.
- CUA has concurrent mouse and keyboard support. The user should be able to switch between the two at almost any point.
- CUA supports graphical representations when graphical capabilities are available.

The CUA interface has evolved using new tools and extensions to the underlying operating systems. CUA is described with two models (Berry & Reeves, 1992):

1 Graphical Model. This model defines the GUI that users interact with during execution of applications. It covers the graphical representation of screen objects.
2 Workplace Model. This model defines an object-oriented view of the interface. It supports the user's focus on manipulating data objects with multiple views and direct manipulation.

CUA has evolved from simply providing a consistent interface across a range of environments to application support for advanced GUI technologies (such as OS/2 Presentation Manager).

Apple Guidelines

The Apple interface has undoubtedly been the force behind the GUI market and the benchmark used by other competing GUIs. Since its introduction through the Lisa and Macintosh computers, users have benefited from the 'desktop' metaphor and have forced other companies to supply GUI interfaces for their operating systems.

Apple have a rigid set of guidelines and provide an programmable 'toolbox' of routines that allow the developers to implement the guidelines successfully. The Apple guidelines allow for innovation in the interface, as was seen when HyperCard was released with its tear-off menu palettes.

The guidelines are covered in (Apple Computer, 1987) as well there is a series of interface updates called the 'Human Interface Notes' that describe any new or changed interface guidelines (sometimes as a result of System Software updates). As an example, Erickson (1990) describes the 'duelling metaphors' problem that HyperCard introduced to the System Software. In this case it was recommended not to use HyperCard style buttons (requiring single mouse clicks) for desktop applications (usually requiring double mouse clicks) as this will lead to user confusion and inconsistencies. Tognazzini (1992) describes intuitive insights and practical technical experiences with the development of the Apple guidelines that capture the underlying principles of graphical user interface design. He also reports on the challenges faced by such guidelines with multimedia, agents, virtual reality, and future technologies.

Generalised Guidelines

Generalised guidelines are attempts to provide non-hardware/software specific interface guidelines. They are usually gathered from diverse sources and presented in a manner that can be applied to specific interfaces environments. Guidelines are based on results from experiments, rule-of-thumb, or principles from human factors and interface design.

Cox & Walker (1990), Brown (1988), and Brown & Cunningham (1989) describes numerous guidelines as well as a providing an effective checklist for the designer to follow and apply. Brown also contains strategies and examples for incorporating the guidelines into the overall development process. Hicks & Essinger (1991) describe 'corporate style' guidelines that have been developed over a number of systems and packaged together as the standard to use for the organisation's subsequent projects.

Marshall *et al* (1987) and Maguire (1990) present a compilation of guidelines for the design of the user interface with the emphasis on the cognitive aspects of interface design as well as the human factors involved in GUIs. The method they use to derive the guidelines is one of 'showing-by-example' which results in related sets of immediately accessible guidelines with concrete examples.

Mayhew (1992) and Galitz (1993) are recent examples of comprehensive sets of guidelines that try to encompass all aspects of interface design with generous illustrations and examples. These range from guidelines for text-based screens through to graphical displays on a full range of computer systems. It is interesting to note that most of the text screen guidelines also apply to graphical displays. Both sets of guidelines are based on sound human factors research and practical design principles.

Specific Guidelines

Guidelines can also be found that detail specific areas of the interface. Such areas may include: menus, icons, colour, typefaces, form filling, navigation, and windows.

Marcus (1986) sets out tested principles on the use of colour for interfaces, in particular which colours should not be used together or in certain areas of the display. Such guidelines are based on proven experimentation and user testing and alleviate the problem for software developers. (See 'Colour' on page 23.)

Often a good source for icons to be used in interfaces is from international and national bodies. One example is the aviation authorities as they have similar problems with providing internationally recognisable icons for services to a wide range of cultures. Modley (1976) and Dreyfuss (1972) provide a rich source of easily recognized icons that may be applicable to metaphors used in interfaces.

2.8 Evaluation and Usability

The evaluation of the interface is undoubtedly one of the most important aspects for applications to be successful with users. Without evaluation, designers would have little indication as to the successful and (more importantly) the unsuccessful interface designs. From a management point of view, the evaluation of an system is important in determining the feasibility of introducing HCI techniques (rapid prototyping, human factors, UIMS) into the development lifecycle.

Evaluation also allows some comparisons in the cost savings for new (and presumably better) interfaces over existing systems (Eaves, 1991). Karat (1993) reports on a possible cost-benefit ratio of 1:100 for projects involving usability evaluation on the interface.

2.8.1 Usability

Usability evaluation is defined as an evaluation of a system that is designed to measure the functionality of the system in respect to the undertaking of users tasks. Such evaluations vary in terms of (Vainio-Larsson & Orring, 1990):

- when they are performed,
- what they attempt to measure,
- how they are performed, and
- whether they are comparative or focused.

Evaluations are normally performed during the development of the system to catch any potential problems early. This also needs attention, as the integration of usability evaluations with system design and development require close cooperation of the entire production team. Hammond & McManus (1991) discuss this problem and identify the critical success factors such as clear evaluation objectives, inclusion of developers in the process, and evaluating all components of the product.

There are numerous methods for evaluation and usually a combination is used to provide a complete evaluation of an interface. Kishi & Kinoe (1991) examine four usability methods to assess their effectiveness and suggest that no single evaluation method can meet all the criteria. Instead, they recommend a formal evaluation method to be used in the early stages, then an observational method later in the development. Other comparisons of usability evaluation methods have produced similar conclusions (Miller & Jeffries, 1992).

Whiteside *et al* (1988) provide a comprehensive review of usability engineering techniques and illustrate how they are used in several situations by providing a framework for the process. Crellin *et al* (1990) evaluate a number of different evaluation methods and speculate that carefully designed interfaces of any type are likely to have similar usability requirements. Their evaluations cover formal, empirical, contextual, and ethnographic approaches to interface analysis.

Israelski *et al* (1989) describe a comparative evaluation between a graphical direct manipulation interface and a corresponding text based interface. In this case, speed and performance was measured on a number of tasks and the results led to new product-specific user interfaces for the company. Plaisant & Shneiderman (1992) evaluated four different touchscreen interfaces that involved scheduling date and time events. From their usability studies they concluded that one style was favoured which was then used in the final product. Happ *et al* (1991) evaluated alternative designs for variable selections lists in the CUA interface. Their recommendations can be used for developments using the CUA interface as well as a general guide to the most effective methods for list selection.

As a general guideline, a usability test has three main ingredients (Potosnak, Nov 1988):

1 real users,
2 real tasks, and
3 real products.

The users should be representative of the typical user of that application and the number should be of optimal size for the anticipated evaluation results. The tasks should represent a whole user task and will allow assessment of the interface for consistency between users conceptual models of the tasks. The usability tests should be performed on real products or working prototypes and not rely on the users imagination of the final product.

Cox & Walker (1990) recommend that a usability test be performed on an existing system before designing a new system. This will allow the designer to show improved productivity in the new interface and gives an understanding of the problems that the new interface is trying to solve.

Usability testing draws on the underlying contributions of experimental psychologists (building HCI's multidisciplinary tag) and the investigative tools and methods used. Lindgaard *et al* (1991) describes a number of usability tests and suggest that experimental methods can fruitfully be applied to various stages of systems development. Their findings do uncover that the more realistic the experimental environment, the less experimental control and conversely, the less realistic the experimental environment, the greater the experimental control. A balance must be achieved to generate results that are applicable and representative of the interface under evaluation. De Waal & Van Der Heiden (1990) also describe an approach for usability testing by embedding the skills of ergonomists and cognitive psychologists in the design process. The term 'macroergonomics' is used to describe this development.

2.8.2 Design for Usability

If a system under design will be subjected to usability evaluations, then it makes sense to try to incorporate and use methodologies in the design that will aid the evaluations. Gould & Lewis (1985) recommend three key principles of design:

1 Early focus on users and tasks,
2 Empirical measurements, and
3 Iterative design.

Designers must understand the users of the system by studying their cognitive behaviour and the real tasks required to be accomplished. Early in the development, users should be tested with prototypes and their performance analysed and feedback used in redesign. There should

also be an iterative process of design, test, and redesign that is repeated as problems with the interface are discovered.

These three key principles were mandatory techniques used in the design and evaluation of the Iris prototype. The focus on users and their tasks leads to a more effective user interface closely matching their expectations.

An extension (Shackel, 1988) to these three key principles, although not used in the design and evaluation of the Iris prototype, include:

1 Participative design, and
2 User supportive design.

Participative design includes end users of the system on the design team to enable a clearer and immediate communications path between the two. User supportive design includes appropriate training, manuals, and on-line help system to assist the user in the operation of the system.

Design of interfaces is not an easy exercise, but following the above recommendations and including some of the key principles will result in positive usability evaluations.

2.8.3 Task Models and Analysis

For effective software design, designers must have knowledge of the types of tasks that the users are attempting, either by physical means or existing computer systems. By using 'task analysis', appropriate models can be formed and used as a basis of the design. However, it can be clearly seen that task models must be true representations of what the users really wish to accomplish, otherwise the initial designs would be flawed. Of course, with consistent prototyping, these early mistakes may be found and rectified.

Potosnak (July 1989) describes the results that task analysis can reveal including the number and proportion of tasks that users perform and the order in which tasks are performed. The data from task analysis can be useful in determining:

- the functions a system should provide,
- how the functions should be implemented, and
- typical task times.

Diaper (1989) points out the problems involved in observing users during task analysis and the various recording technologies that can be used. The major problem is selecting *what* to observe and deciding which parts of the observation are not relevant to the task and what level of detail is required to capture the task steps. Other decisions include; who, when, and how often the observations will be recorded.

Waddington & Johnson (1989) describe an enhanced task analysis method which shows how the task behaviour is supported by the interface behaviour. The 'object-based' notation used also shows how task models can be used to suggest alternative interface designs which can be predicted to be more usable.

2.8.4 Discount Usability Evaluation

Usability evaluations can become expensive and quite difficult to manage which may endanger its existence in the development cycle but 'discount' usability testing has proven to be just as effective (Nielsen, 1989b). This involves selecting a small number of low cost evaluation

methods in a situation of resource constraints. The following sections describe evaluation methods that may be used. Their inclusion here reflects each methods ability, with respect to resource requirements, in generating effective usability results.

Thinking-Aloud Method

One of the most interesting and cost-effective human factors methodologies for interface evaluation is the 'thinking-aloud' method (Lewis, 1982) also called 'verbal protocols'. Users are asked to perform certain well-defined tasks using a software package. They are also instructed to keep a running commentary of their thoughts. This includes what they are doing, any problems they encounter, and what they expect the computer to do next.

This invaluable feedback gives the designers insight into the user's model of the system and its compatibility with the designer's model. Using an observer or video, the method can not only capture that users are having problems but *why*. It also provides useful information on how users approach each task and can work quite satisfactorily with early prototypes.

Jorgensen (1989) has researched the effectiveness of using the thinking-aloud method in practice and has made the following promising recommendations:

- highly suited for debugging user interfaces,
- applied to complete systems, prototypes, and paper mock-ups,
- applied at any time during the development,
- used by designers with little or no training in human factors,
- complete evaluation takes about 2–3 person weeks,
- three persons are in general appropriate for each test,
- logging, audio or video recording is required,
- designers can be observers (if they can exhibit self-control), and
- conduct informal interviews after the test.

As can be seen from this, the thinking-aloud method is a very useful instrument for designers of user interfaces and requires modest resources for effective evaluation results. Wright & Monk (1991) also report on the success of the thinking-aloud method in the cost-effective evaluation of prototype interfaces using set tasks.

However, the thinking-aloud method does have some disadvantages including the following:

- the user's reluctance to participate fully,
- an interpretation of user's comments,
- an informal procedure that could be open to biases, and
- a slowing of the user's performance.

Scenarios

Scenarios are small tasks that the user may typically perform on a regular basis when interacting with the software. This allows the designer to develop a prototype that supports the execution of the scenario (so that user testing will succeed) but eliminates the complexity of the full system.

Developing effective and small scenarios that may cover the bulk of the typical tasks a user may perform enables the designer to get quick and frequent feedback. The scenario may reduce the level of functionality but may instead emphasise a specific interface style under eval-

uation. The prototypes used may also be developed more quickly so as to only simulate the interface being tested with the scenario. Scenarios have proved to be effective in feeding back information to the development team (Tyldesley, 1988).

Heuristic Evaluation

There are currently of the order of one thousand usability guidelines which may intimidate and hinder their use for developers (Nielsen, 1989b). These can be sensibly cut down to include only the major guidelines that present the largest proportion of problems in user interfaces. Such a list from Molich & Nielsen (1990) is listed in Table 4.

Table 4: Usability Heuristics

Usability Heuristic
Simple and Natural Dialogue
Speak the Users Language
Minimise the User's Memory Load
Be Consistent
Provide Feedback
Provide Clearly Marked Exists
Provide Short-cuts
Provide Good Error Messages
Error Prevention

Each heuristic is applied to the user interface and challenged for validity. The use of such heuristics does require some experience with the principles and it may be the case that an outside consultant be used to do the evaluation. If this is not possible, then using a number of different people to perform the evaluation may highlight sufficient problems in the interface.

Interface Monitoring

Interface monitoring involves the generation of a transcript of the actions that a user performs whilst interacting with the software. This is usually carried out automatically by the software which would write an audit file whenever the user performed an appropriate action. The largest problem with this technique is the analysis of the vast amount of audit information that is generated. Usually a separate tool is required to provide some form of summary of the audit information so that conclusions can be made of the interface (Rees & Iannella, 1990b). Siochi & Ehrich (1991) developed a tool that provides an indication of the repetition of the user actions from a session transcript that may highlight potential user interface problems.

Apart from the application developer including the audit generation in the code for the software, some tools do exist for the automatic generation of the transcripts. Macleod (1989) describes a tool that can be used for HyperCard stacks to generate and view the interaction record. Chen (1990) outlines the changes made to the X Windows System (R4 version of the X11 Intrinsics) that with a small number of application code changes can take full advantage of the inbuilt monitoring mechanism. The developer can also specify which tasks to monitor.

Questionnaires

One of the least expensive usability evaluation methods is providing short questionnaires to users about the performance and functionality of a system. Questionnaires require a lot of planning but do allow for the collection of data from large numbers of users. The formats of the questions should help gather as much quantitative information as practical and should be worded to elicit such responses (Perlman, 1989).

2.9 The Future

Shneiderman (1990) describes some of the goals and application areas for future research and development in HCI. He mentions UIMS as one of the key tools and underlying technologies in developing these future applications. Another key area is in providing flexible tools for finding information in a non-structured way (resource discovery). This type of 'universal function' is also mentioned by Perlman (1992a) as an area that can be exploited in future interfaces.

Designing interfaces for the future will involve powerful tools and a greater use of intelligent 'agent' interfaces. Challenges to designers will include focusing on innovation. Restricting design to evolutionary improvements will not be able to deal with new technologies such as virtual reality. The history of HCI has included revolutionary steps forward which must be encouraged and evaluated (Fischer, 1989).

Olsen *et al* (1993) describes three new research issues:

1 Malleable interfaces that provide users control of the form and content of computing tools in their end-user computing environment.
2 Interface development environments that go beyond simple implementation to support and integrate the range of activities involved in developing interfaces.
3 Supporting technologies supplied from other disciplines that impact on the development of user interface software.

New and enhanced interaction techniques such as the use of eye movements (Jacob, 1991) and Datagloves (Wilson & Conway, 1991) as well as advances in new and faster hardware provide exciting challenges for designers. Developing new interface toolkits, new metaphors, new evaluation methods, and new techniques will provide HCI researchers directions and areas to address. An example includes the requirements for pen-based user interfaces (GO Corporation, 1992) with the creation of new metaphors.

Greater resources are required by HCI practitioners, including case studies of successful implementations of the techniques discussed in this chapter, to enable effective user interfaces to be design and evaluated. This is one area, via the development of the messaging user interface reference model, that the Iris prototype will contribute to the HCI community.

Chapter 3

Message and Directory Standards

3.1 Introduction

This chapter introduces the concepts, functionality, and applications of electronic messaging systems. A discussion of the architecture and various standards of electronic mail systems is presented. This includes areas such as addressing, multimedia messages, security issues, integration with various gateways, and electronic mail applications.

The major focus is a comprehensive overview of the X.400 Message Handling Systems and X.500 Directory Services international standards. These standards are thoroughly discussed to gain an appreciation of their complexity and size. The key issues of each standard are presented to enable a greater understanding of the original problems and features lacking in X.400 and X.500. A review of the current research and applications applicable to both standards is also outlined.

3.2 Electronic Mail

Electronic mail broadly covers the area of message communications across a heterogeneous network of disparate computer systems. Electronic mail has been defined as:[1]

> *Electronic Mail is the generic name for non–interactive communication of text, data, images or voice messages between a sender and designated recipients by systems utilizing telecommunications links.*

This definition covers the computer-based message systems and the full range of communications systems such as voice mail, telex, and facsimile. It is interesting to note that this definition perceives electronic mail as being 'non–interactive'—emphasising its lower layer view of electronic mail. Contemporary impressions, including this research, place great importance on the interactive capabilities of electronic mail.

---

1. Source: Electronic Mail Association, May 1986.

Electronic mail has a number of advantages over its corresponding paper-based cousin:

- Electronic mail is asynchronous which enables quick message delivery.
- Message transmission uses a 'store-and-forward' technique which does not require the recipient's host machine to be continuously available.
- The nature of electronic mail allows it to be easily integrated into office applications and message data can be readily used in other applications.
- Electronic mail is cost-effective and empowers the user, allowing one message to be just as easily sent to 100 recipients as one recipient with little or no duplication of resources.

Although there are no clear disadvantages, it can be seen that electronic mail is an effective tool for user communications and provides many advantages to office and research environments. Vervest (1985) provides a comprehensive discussion on the annals of electronic mail and communications including the technical details of message transfer over various media. Users expect mail systems to provide common services that they are familiar with from elsewhere (eg letters, telephone) and that are provided in a more convenient way (eg faster delivery) with additional services (eg multimedia) that are not currently provided (Jakobs & Karabek, 1992).

3.2.1 Basic Concepts

Electronic mail relies on an underlying transport mechanism that provides the transmission of messages between computer systems. This network transport layer may be any reliable protocol such as X.25 or TCP/IP which usually does not interest the end user. A *mail server* usually resides on top of the transport layer and provides the routing of messages to the recipients mail server. On top of this, electronic mail consists of a user *mail agent* for the management of mail messages. The mail agent provides the following functions:

- create messages,
- read messages,
- reply to messages,
- forward messages,
- store messages, and
- retrieve messages.

This mail agent provides the necessary facilities for a user to use in sending messages to other users, the reception of messages, and the management of the messages. This is shown graphically in Figure 22.

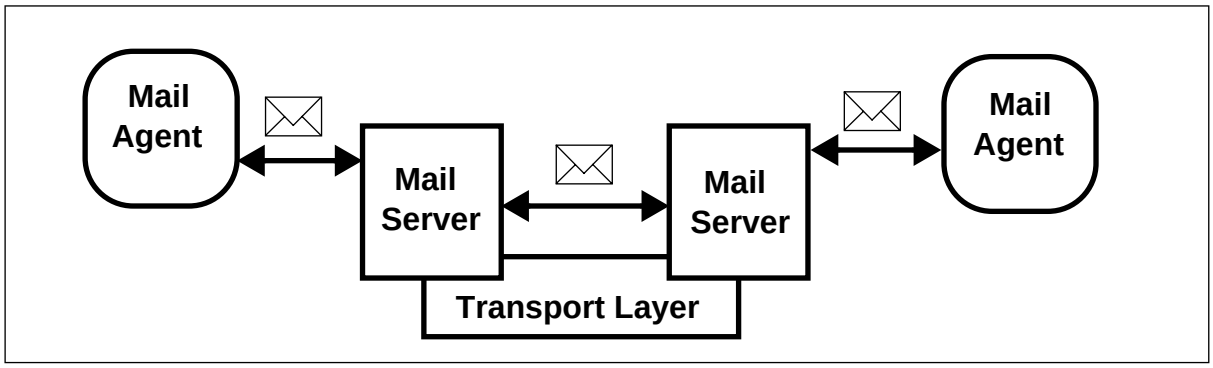


Figure 22: Electronic Mail System

The structure of the mail message is similar to that of a standard office memorandum consisting of a header and body part. The header part is a series of fields including (but not limited to):

- recipients,
- carbon copy recipients,
- originator,
- subject, and
- date.

The body part usually consists of the main text of the message. This is shown graphically in Figure 23.

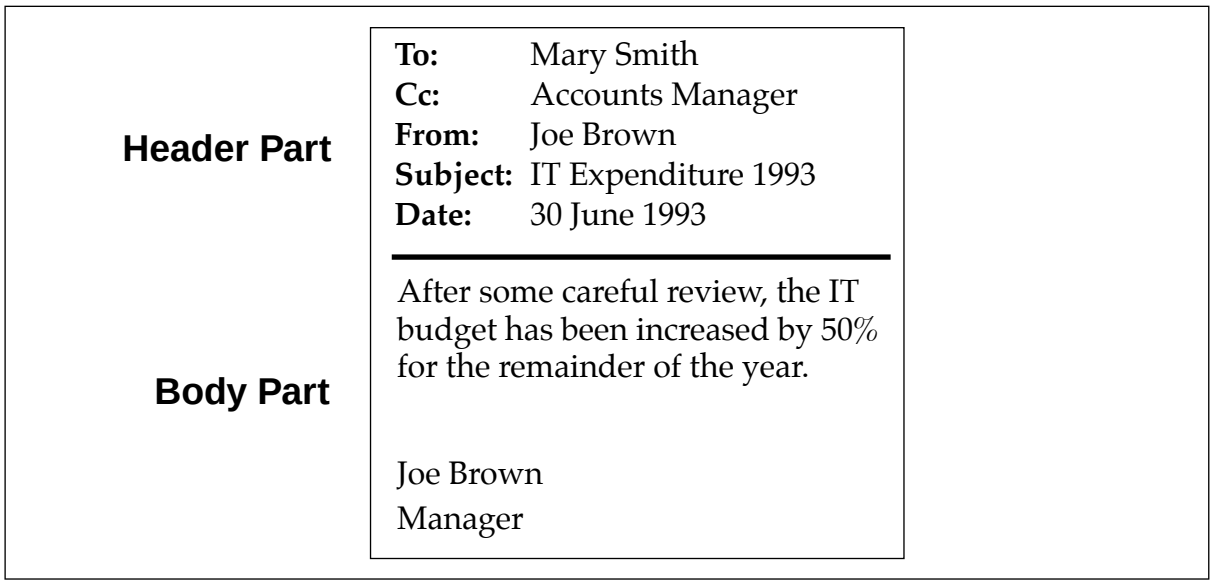


Figure 23: Mail Message Structure

3.2.2 Architecture

Electronic mail systems are based on the client-server paradigm. A mail user does not deal with direct communication with the intending message recipients. Instead, the user will communicate with a server which, like the postal office, deals with the transportation, routing, and delivery of the message. The client-server paradigm provides another key feature of *store–and–forward* delivery. This allows the server to hold delivery of a message until the next server on-route is available on the network. Hence, delivery does not depend on all servers in the route to be operational or temporary network outages. Once communications are restored the message can be delivered to the next server until the final destination is reached. At this point the message will be available to the client mail agent software.

By following this architecture, heterogeneous electronic mail systems can be easily integrated. (See '3.2.6 Integration' on page 53.) Cini (1990) reports on such successes between microcomputer and mainframe electronic mail systems. Structuring the message environment adds value to the internetworking opportunities (James & Churcher, 1988) as well as using an object-oriented message paradigm (Kuo, 1988).

3.2.3 Standards

Standards can be classified as de facto or de jure. In electronic mail, the X.400 Message Handling Systems is the de jure standard and is fully discussed in section '3.3 X.400 Message Handling Systems' on page 55. The are a number of de facto standards in electronic mail, the most popular being the Simple Mail Transport Protocol (SMTP) which is widely used on UNIX systems. Others include UUCP, Bitnet and JNT mail. Although there is a competing rivalry between these standards (Hayes, 1992b), the de facto standards are still in force due mainly to the large installed base and relatively easy implementation. It will be only a matter of time when the de jure standards will be prominent, but until then the de facto standards are offering gateways into the de jure standards.

SMTP is defined in RFC 821 (Postel, 1982) and is a relatively simple protocol for the transmission of messages. Using a small set of commands, SMTP can deliver mail message to any other server using the same protocol over a network. The format of the messages that can be transmitted by SMTP is defined in RFC 822 (Crocker, 1982) and is a text-only format consisting of various header fields and a body part. A comprehensive list of electronic mail systems can be found in (Lewis, 1992; Plattner & Lubich, 1989) with a discussion of their implementations.

The many standards has led to some confusion and paralysis with vendors and users. The promises of 'interoperability through standards' cannot always be delivered. Marshak (1993) examines the current status of the electronic mail standards and the various approaches to assuring a reliable mail-based system for businesses. He concludes by stating that it is no longer a question of *if* de jure standards are the answer but *when*.

Addressing

An important part of the RFC 822 standard is the syntax used in addressing the message to recipients. The address of a recipient is divided into the recipient's name, followed by the '@' symbol, host computer name, and the domain names. The latter two are dot-separated tokens.

The recipient's name is usually the user's account identification name. The host computer name is the name of the computer that the recipient has an account on. The domain names represent a logical view of the global network that the recipients machine is a part of. It usually consists of four tokens representing the:

- department,
- organisation,
- organisational class (or sub-network), and
- country.

For example, a (fictional) recipient, Fred Smith, who works in the Law School at Bond University who has an account 'fred' on the computer 'surf' would have the following address:

fred@surf.law.bond.edu.au

One thing that is clear from this is that the addressing scheme used may benefit the mail server in determining the route to take to deliver the message, but it does not provide a user with a genial or predictable address. The recipient account names on computer systems also vary greatly and there is no standard for such naming.

For example, other possible addresses for Fred Smith include:

smithf@surf.law.bond.edu.au
freds@surf.law.bond.edu.au
smith@surf.law.bond.edu.au
rambo@surf.law.bond.edu.au

The latter example is the case in which the recipient has used a common personal nickname.

System administrators can also setup alias entries for recipient addresses in an attempt to provide more sociable addresses or to hide parts of the address. The mail server would then lookup and expand the alias address into the real address. For example, the following are two common address alias entries for Fred Smith:

fred_smith@bond.edu.au
f.smith@bond.edu.au

Although this maybe easier to remember, it still does not provide any consistent method for addressing messages across organisations.

Another type of recipient address is used for distribution lists. The same syntax is used but the recipient account name is in fact an alias to a number of other (human) recipients or even other distribution lists. Upon receipt of such a message, the mail server would expand the distribution list address and send the message to the individual members of that list. This is useful if a group of recipients wish to receive messages on the same topic which only requires addressing to one recipient (the distribution list).

3.2.4 Security

The security for electronic mail is severely lacking in the de facto standards. The authentication of users is usually via the login process to the host computer. After that point, there is no guarantee that mail messages will reach the destination without others 'eavesdropping' or that the sender of the message is valid ('masquerading'). For electronic mail to be totally accepted (especially by the commercial users) there must be some form of guaranteeing minimum levels of security.

New methodologies have been developed to enhance the de facto standards such as RFC 1113 (Linn, 1989) and the de jure standards all address the security issues as a central key factor. These security issues provide confidentiality, integrity, authenticity and in some cases, non-repudiation of messages by using forms of cryptography and secure-key management.

Bishop (1991) describes how such security methods can be achieved in mail agents by providing a tool to encrypt and decrypt messages which are treated as 'normal' messages by the mail server. Such *Privacy Enhanced Mail* (PEM) utilises the Public Key security of X.500 Directory Services. Mitchell (1989) outlines some of the problems in the use of security for multi-destination messages and a possible solution.

3.2.5 Multimedia

Electronic mail messages used in the de facto standards have been limited to a single text-only body part. Multimedia documents have been with us for quite some time and it is only natural that users would require that such documents could be transmitted as electronic mail messages. Multimedia documents could include, text, images, facsimile, voice, binary data, and video.

Early attempts at experimental multimedia mail system (Katz, 1984; Postel *et al*, 1988) have produced successful and encouraging results. The user interface for such systems is of key importance as interaction with multimedia documents requires great care in their construction (Yankelovich *et al*, 1988) and spatial layout (Garcia-Luna-Aceves, 1986). Diamond (Forsdick *et al*, 1984; Thomas *et al*, 1985) is another example of a multimedia message system that raises the question of what impact multimedia capability will have on personal electronic communications.

Models for multimedia document exchange (Skogseth & Verne, 1986) as well as proposed multimedia message protocols (Garcia-Luna-Aceves & Poggio, 1984) have assisted the de facto standards in formulating extensions to encompass multimedia message transmissions. One of the most promising de facto standards is the Multipurpose Internet Mail Extension (MIME) described in RFC 1341 (Borenstein & Freed, 1992). MIME encodes the multimedia parts of a message into text format, thus enabling existing systems to transmit the message. The opposite occurs at the recipient's ends, with the encoded text converted to the original multimedia part. MIME's simplicity has lead to many successful implementations such as Pine (Siebel *et al*, 1992) and Metamail (Borenstein, 1991).

The de jure standards allow for multimedia messages by separating the body of messages into multiple parts. Each body part is of a different multimedia type but would be viewed as a single message by the user. One problem arising from this is the delivery of multimedia messages to recipient computers that cannot display or use the multimedia type. In such cases, the body type would be converted (if possible) to a different form or the recipient would be informed that the body part could not be displayed.

3.2.6 Integration

With many disparate electronic mail systems and no clear and easy migration solution to a single de jure standard, the alternative is to provide gateways between the heterogeneous systems. This allows the use of the existing systems and only requires the addition of a gateway system. The gateway would convert or repackage the message into a common format that could be transmitted over a different network to a recipient using a dissimilar electronic mail system. Figure 24 shows a mail gateway that is servicing the messages from different mail agent systems running on two heterogeneous networks.

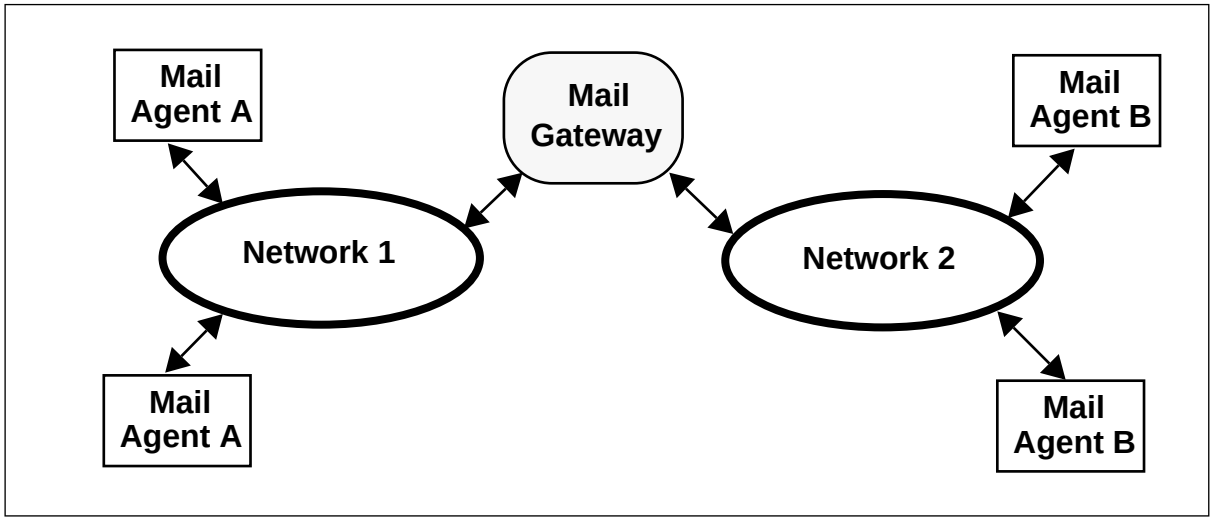


Figure 24: Electronic Mail Systems Integration

On larger networks, a backbone network is used with heterogeneous sub-networks interconnected. The users of each sub-network may still use the mail tools that have evolved to suit their needs with gateways onto the backbone network. This 'lowest common denominator' requirement for successful backbone networks is summarised by (Hu, 1988):

- Use of a family of compatible gateways supporting translation services.
- Support a variety of gateways that suit the capabilities of different machines.
- Obey the invariant that messages circulate in their native format on the backbone and are translated when they exit.

Hamer & Heilmann (1988) proved that integrating five different mail systems on the Citicorp Global Telecommunications Network and providing a uniform and seamless environment for users is possible using gateways to de facto standards. With increased competition, software corporations are offering gateways for their messaging systems to other platforms. Microsoft Corporation (1992) offer strategic migration plans for their products with the final phase being based on a fully de jure standards solution. Fortunately the underlying architecture for integration is available, the real concern is the resources required for the investment and the consistency for access to the gateways (Martin, 1992). Eglowstein & Smith (1993) provide a summary of common microcomputer-based electronic mail systems and the integration paths that are available to build a multi-platform internetwork mail system using external gateways.

3.2.7 Application Programmers Interfaces

Electronic mail application programmers interfaces (APIs) are methods that can be used by programmers to access mail system services. This may include the transportation of messages, the directory of addresses, or the storage of messages. Mail APIs also allow common applications (eg word processors) to submit, receive, or manipulate mail messages. This has led to the term 'mail-enabled' applications and includes applications that can simply send a mail message to complex groupware and scheduling applications.

Mail APIs have allowed developers a simpler methodology to produce mail-enabled applications by providing a multi-platform interface to various systems. The developer uses a consistent high-level and streamlined syntax to access the mail services regardless of the underlying protocol communications.

Corporations have published their API specifications in the hope that they may become de facto standards and to provide a competitive advantage. Marshak, David S (1992) believes that, in the short-term, this may create problems as to which API to use. In the long-term, the API mail services—through customer demand—will be accessible by any of the API sets.

The major players for APIs on the market include the following:

- MAPI is Microsoft's Messaging Application Programming Interface available for Windows 3.
- VIM is the Vendor-Independent Messaging jointly backed by Apple, Borland, Lotus, and Novell as a cross-platform interface.
- OCE is Apple's Open Collaboration Environment for the Macintosh.
- MHS is Novell's Message Handling Service.
- XAPI is the X.400 API Association's interface for X.400 services.

A complete survey of these APIs can be found in (Marshak, David S 1992b). Other APIs are surveyed in (King, Steven S 1992)

3.2.8 Applications

Electronic mail is in widespread use throughout organisations and is undoubtedly an effective means of practising work-place communications. Keeping abreast of electronic mail service developments is commonplace as organisational staff become aware of these administrative advantages (Updegrove *et al*, 1990) including the integration of electronic mail to other information resources (eg bulletin-boards and conferencing systems (Kille, 1984)). Studies have found that electronic mail is viewed as the medium that has significant impact on how individuals communicate in an organisation (Adams *et al*, 1993).

Apart from the messaging-centric applications like electronic mail systems, scheduling and calendaring, there are the 'message-enabled' applications. These applications have direct access to an electronic mail system and its use is transparent to the user. Word processing applications that can automatically enclose documents to messages and groupware applications for multi-user access to documents are two examples.

Electronic mail applications traditionally present all messages to users and can suffer from 'information-overload'. Extensive research has been provided into methods of filtering messages and adding intelligent agents to electronic mail systems to act on a set of user-defined rules (Pollock, 1988; Lutz *et al*, 1990). Recent mail–enabled applications (eg BeyondMail[2]) provide a 'Message Clerk' agent that processes incoming messages based on a set of 'when-if-then' rules and carries out any actions specified on the messages. Marshak, Ronni T (1992) provides a summary of BeyondMail's features including a description of the structure of the rules-engine.

One of the many potential applications for electronic mail is work-flow automation (Reinhardt, 1993). Work-flow packages digitally represent business processes (eg purchase-orders and invoice processing) that involve the routing of electronic documents, boosting efficiency and adding intelligence to data distribution.

3.3 X.400 Message Handling Systems

The impetus for a de jure standard for Message Handling Systems (MHS) emanated from the inadequacy of the de facto standards in the following key areas:

- a common message structure,
- limited to text-only messages,
- lack of status reports, and
- arcane addressing requirements.

In 1984, the International Telegraph and Telephone Consultative Committee (CCITT) released the X.400 series recommendations on MHS. This was further developed and a new release in 1988 also coincided with the Message Oriented Text Interchange System (MOTIS) standard of the International Standards Organisation (ISO). Both standards now reflect the same recommendations with respect to message handling systems. The X.400 series of recommendations (CCITT, 1988) is listed in Table 5.

These standards are internationally agreed and in the public domain meaning that a messaging infrastructure has been established providing a 'future proof' means for MHS.

[2.] BeyondMail is a registered trademark of Beyond Incorporated

Table 5: X.400 MHS Recommendation Series

CCITT	Name of Recommendation	ISO
X.400	MHS: System and service overview	10021–1
X.402	MHS: Overall architecture	10021–2
X.403	MHS: Conformance testing	
X.407	MHS: Abstract service definition conventions	10021–3
X.408	MHS: Encoded Information type conversion rules	
X.411	MHS: MTS—Abstract service definition and procedures	10021–4
X.413	MHS: MS—Abstract service definition	10021–5
X.419	MHS: Protocol specifications	10021–6
X.420	MHS: Interpersonal messaging system	10021–7

3.3.1 MHS Functional Model

In the MHS model, a user is either a person or a computer process that interacts with the MHS. The users are either direct users and engage in message handling by direct use of MHS or are indirect users and engage in message handling through an external communication system. Such systems include the physical delivery system.

A user is referred to as either an originator (when sending a message) or a recipient (when receiving a message). An originator prepares messages with the aid of a user agent application. A user agent (UA) is an application process that interacts with the message transfer system (MTS) or a message store (MS), to submit messages on behalf of the user. The MTS delivers the submitted messages to the recipient's UA, access unit (AU), or MS. The MTS can be instructed to and can return delivery and non-delivery notifications to the originator. The UA can accept delivery of messages directly from the MTS, or it can use the capabilities of the MS to receive delivered messages for subsequent manipulation by the UA. The MTS comprises a number of message transfer agents (MTAs). Operating together, in a store-and-forward manner, the MTAs transfer messages and deliver messages to the intended recipients.

Access by indirect users of MHS is accomplished by access units (AUs). The AU enable non-MHS users to send messages to MHS users. Delivery to indirect users of MHS is also accomplished by AUs. One example is the physical delivery access unit (PDAU) which integrates to the postal service. The message store (MS) is an optional general purpose capability of MHS that acts as an intermediary between the UA and the MTA. The MSs primary purpose is to store and allow retrieval of previously delivered messages and allows message submission from a UA. The collection of UAs, MSs, AUs and MTAs is called the message handling system (MHS). A functional view of the MHS model is shown in Figure 25.

Direct users interact with UAs for message processing purposes including the creation, retrieval, or storage of messages. A UA can be implemented as a computer application running on a workstation or a computer process requiring no human intervention. The X.400 series does not enforce nor recommend any human interfaces for the UA.

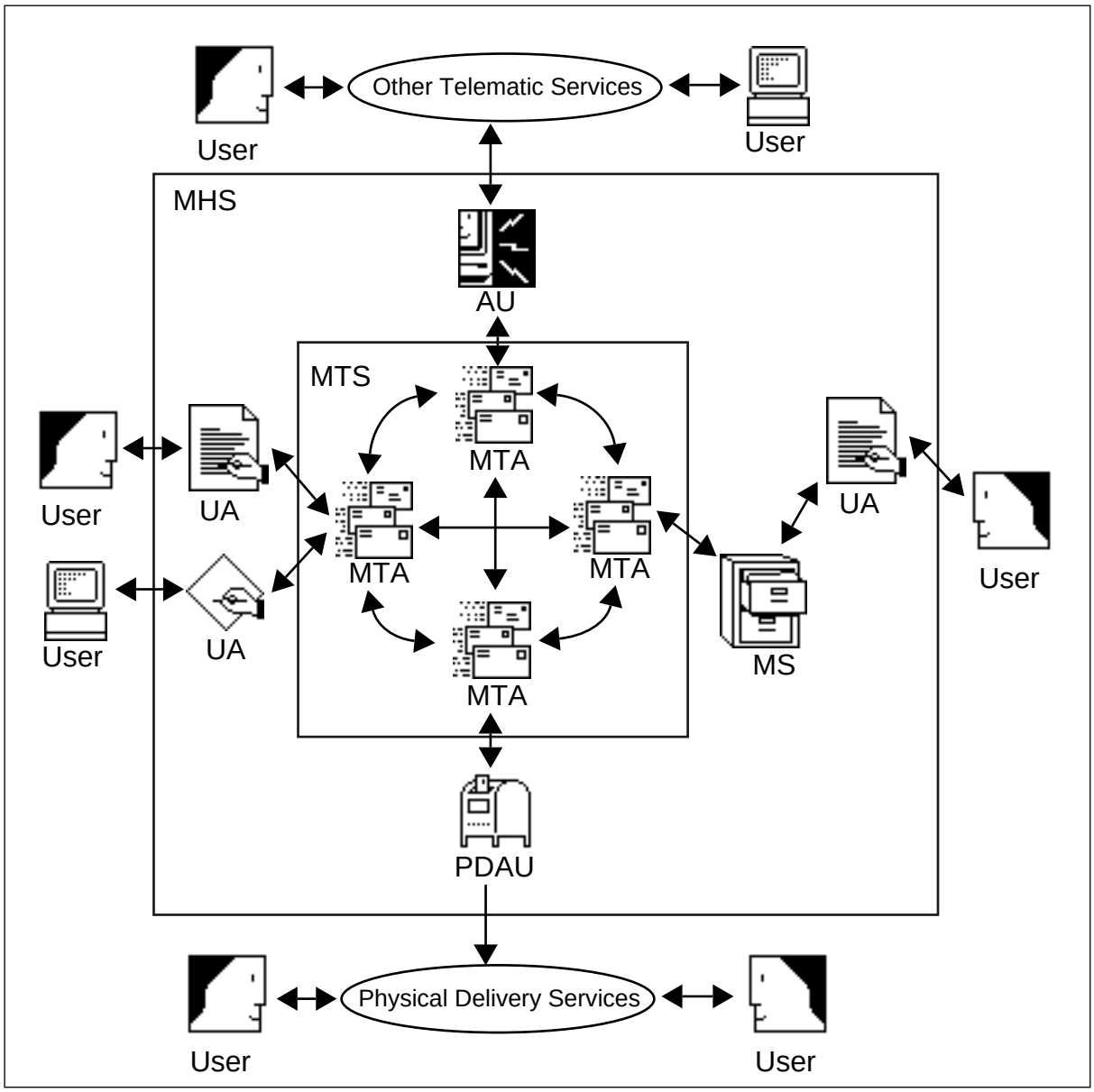


Figure 25: MHS Functional Model

The UA and MTA can be implemented as a physically separate system or co-located on the same system. In either case, the UA and MTA can interact directly to perform MHS services. The UA can also interact with a MS in which case the MS will communicate with the MTA.

UAs are grouped into classes based on the type of message content they can process. The MTS provides a UA with the ability to identify its class when sending messages to other UAs. UAs within a given class are referred to as cooperating UAs since they cooperate with each other to enhance the communication amongst their respective users. UAs can belong to several UA classes if they can support more than one type of message content.

The MTA provides the application independent store-and-forward message routing service. A set of cooperating MTAs providing this message exchange service define the message transfer system (MTS).

There are two methods of interactions between MTAs and UAs and/or MSs:

1 The *submission* interaction is the method by which an originating UA or MS transfers to an MTA the content of a message. The submission includes an envelope containing the information that the MTS requires to provide the requested services.
2 The *delivery* interaction is the method by which the MTA transfers to a recipient UA or MS the content of a message. Also delivered is the envelope containing information related to delivery of the message.

Responsibility for the message is passed between the MTA and the UA or MS during the submission and delivery interactions. The transfer of the message starts at the originator's MTA, and each MTA transfers the message to another MTA until the message reaches the recipient's MTA. The message is then delivered to the recipient UA or MS using the delivery interaction.

The *transfer* interaction is the method by which one MTA transfers to another MTA the content of a message. Also transferred is the envelope containing the information related to the operation of the MTS plus information that the MTS requires to provide the services requested by the originator of the message.

MTAs will transfer messages containing any type of binary coded information and will neither interpret nor alter the content of messages except when performing a conversion. In the latter case, the message may require to be converted if being delivered to a UA that would be unable to display or process the message. The originator also has an option to indicate that no conversion be applied to the message if such a conversion would render the message void.

Notifications in the MTS include delivery and non-delivery notifications. If a message cannot be delivered by the MTS, a non-delivery notification is generated and sent to the originator in a report. Also, the originator can ask for acknowledgement of successful delivery through use of the delivery notification service when submitting the message.

Since UAs can be implemented on a wide variety of systems, including personal computers and workstations, the MS is used to complement a UA in such situations when the UA may not be available. The MS provides a more secure and continuously available storage mechanism to receive messages on the UAs behalf. The MS retrieval functions provide the users with basic message retrieval capabilities applicable to messages of all types. The MS does not provide a shared MS capability, only to individual users.

When using an MS, all messages destined for the UA are delivered only to the MS. The UA, if on-line, will be alerted as to the arrival of new messages. Messages sent to a MS are considered delivered from the MTS perspective. In this case, and when a UA submits a message to the MTA, the MS is in general transparent to the user.

Users are provided with the following MS capabilities, based on various criteria, to access messages currently held in the MS:

- counts and lists of messages,
- fetching messages,
- forwarding messages, and
- deleting messages.

The MS can be located with respect to the MTA in a number of ways:

1 the MS can be co-located with the UA,
2 co-located with the MTA, or
3 stand-alone.

Co-locating the MS with the MTA offers significant advantages which will probably make it the predominant configuration.

An access unit (AU) is a functional entity associated with an MTA that provides intercommunication between the MHS and another systems or services. Figure 25 shows access to other communication systems and services, via a generic AU between the MHS and telematic services. Also shown is a physical delivery access unit (PDAU) to allow for physical delivery of MHS messages to recipients without terminal access to MHS.

The value of MHS can be increased by allowing for the physical delivery (PD) of messages originated within MHS to recipients outside of MHS. In some cases this includes the return of notifications from the physical delivery service (PDS) to an MHS originator. One area still under investigation is the ability for the origination of messages in the PD service for subsequent submission to MHS through the PDAU. The capability of intercommunication between PD and MHS is an optional capability of MHS and all users of MHS will have the ability to generate messages for subsequent PD. An example of a PDS is the postal services which is operated by a management domain that transports and delivers physical messages comprising an envelope and its content.

A physical delivery access unit (PDAU) converts an MHS message to physical form, a process called physical rendition. This may include the printing of a message, addressing, and its automatic enclosure in a paper envelope. The PDAU passes the physically rendered message to a PDS for further relaying and eventual physical delivery.

Other forms of AUs include:

- Teletex access via the telematic agent (TLMA).
- Telex access via the telex access unit (TLXAU) and public telex access unit (PTLXAU).

Other types of access units (eg facsimile and videotex) are for further study.

The basic structure of messages transferred in the MHS consists of an envelope and content (See Figure 26). The envelope contains information required by the MTS to transfer the message and the content is the information that the originating UA requires to be delivered to the recipients. The MTS neither modifies or examines the message content, except for conversions. The message content is specific to the MHS application. An example is described in section '3.3.4 Interpersonal Messaging System' on page 63.

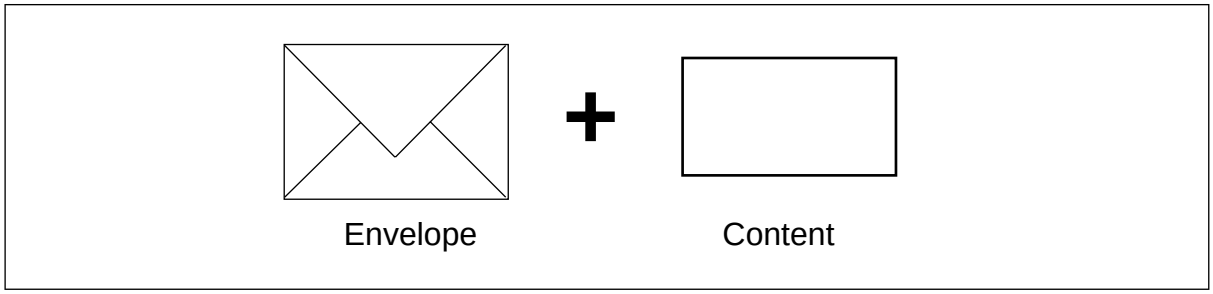


Figure 26: Basic Message Structure

3.3.2 Management Domains and Security

To aid in the implementation of a global message handling environment (MHE), the MHS is further grouped into management domains (MD) with each MD being a collection of at least one MTA, zero or more UAs, zero or more MSs, and zero or more AUs. An MD managed by a Public Telecommunications administration is called an Administration management domain (ADMD). An MD managed by an organization other than a Public Telecommunications administration is called a private management domain (PRMD). Figure 27 shows the relationships between the MDs and the MHS.

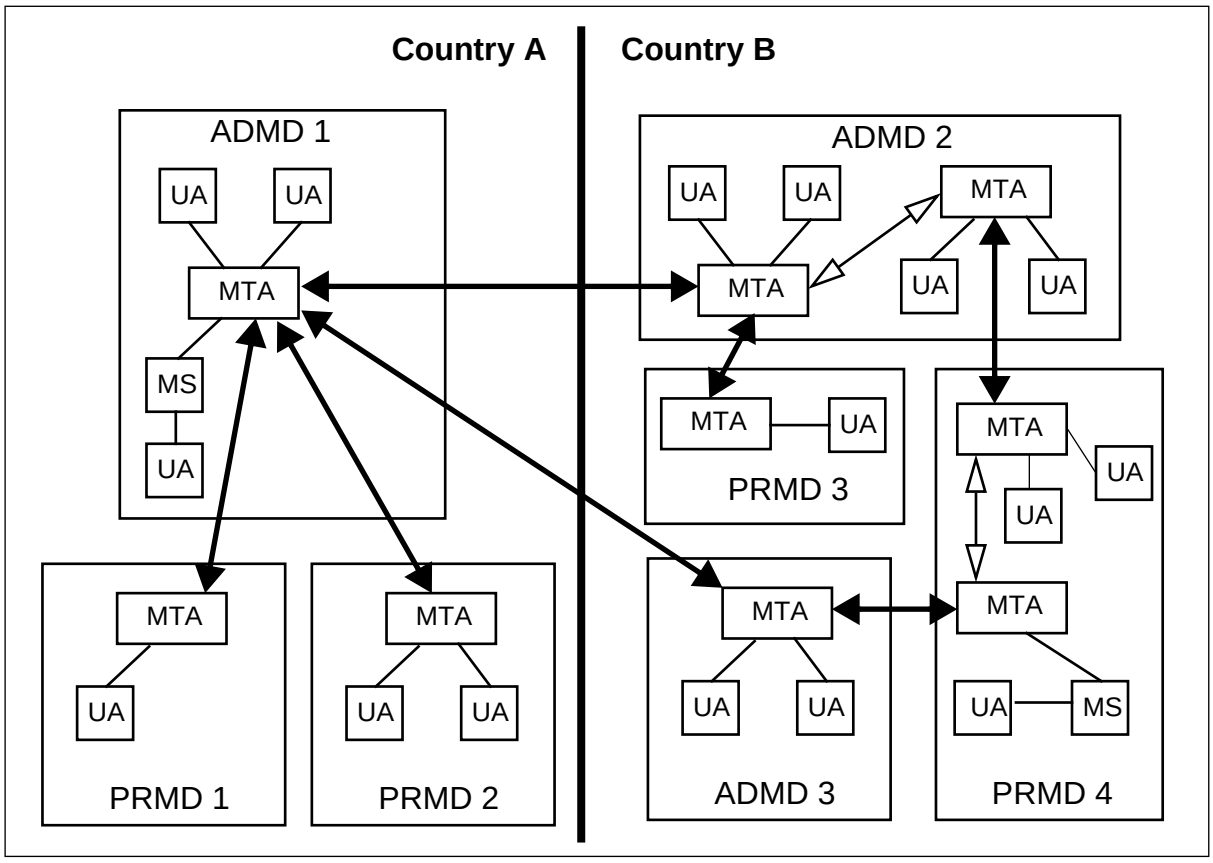


Figure 27: MHS Management Domains

In one country, one or more ADMDs and PRMDs can exist and are characterized by their provision of relaying (on an MTA to MTA basis) and messaging functions between other MDs. A PRMD is considered to exist entirely within one country and can access one or more ADMDs and PRMDs only in that country[3]. To transfer a message to another country, an PRMD must use the services of an ADMD which can interact with other countries ADMDs.

In addition to ensuring that the PRMD properly provides the message transfer service, the ADMD is responsible for ensuring that the accounting, logging, quality of service, uniqueness of names, and related operations of the PRMD are performed correctly. The name of a PRMD can be either nationally unique or relative to the associated ADMD in which case the PRMD can have more than one name.

[3] The ISO standard differs in this respect by allowing PRMDs to cross country boundaries.

The distributed nature of MHS and the acceptance in the financial business community makes it desirable that mechanisms are available to protect against various security threats that can arise. The nature of these threats include the following:

- Invalid user access.
- Masquerading by an impostor into revealing sensitive information.
- A genuine message which has been modified by an unauthorized agent while it was transferred through the system.
- Recorded messages to be replayed to the recipient at a later date.
- Analysis of message traffic between MH users by an eavesdropper.
- Repudiation of messages by one of the participants denying involvement in the communication.
- Security level violation if a MD within MHS employs different security clearance levels.
- Unauthorized modification of the routing information could lead to messages being mis-routed or even lost
- An unauthorized agent could make a copy of a deferred delivery message and send this copy to the intended recipient while the original was still being held for delivery in the MTA.

The X.400 recommendation defines two aspects to security in message handling:

1 *Access security management* covering the establishment of an authenticated association between adjacent components in the MHS (eg between UA/MTA and MTA/MTA).

2 *Secure messaging* covering the application of security features to protect messages in the MHS in accordance with the enabling of various components to verify the origin of messages and the integrity of their content. Secure messaging also prevents unauthorized disclosure of the message content.

Many of the secure messaging functions provide an originator to recipient service. They require user agents with security capabilities and do not require the use of a MTS with security features. The MS also uses secure messaging services but in general, the MS is transparent to security features that apply between the message originator and recipients.

3.3.3 Naming and Addressing

The principal entity that requires naming in a MHS is the originator and recipient of messages. These may be human users or entities such as distribution lists (DLs) representing members of a cohesive group. Users of the MHS and DLs are identified by Originator/Recipient (O/R) names. O/R names are comprised of directory names and/or O/R addresses.

Either or both components of an O/R name can be used as the recipient of a message. If only the directory name is present, the MHS will access a directory to attempt to lookup the recipient's O/R address. The O/R address will then be used to route and deliver the message. If both components are given, the MHS will use the O/R address given. If the O/R address is found to be invalid, the directory name will be used as above.

Users and DLs can be identified by a directory name that must be looked up in an X.500 directory to find out the corresponding O/R address. (See '3.4 X.500 Directory Services' on page 73.) The user can access the directory system to find the O/R address of a user or use the directory name (in the recipient list) and have the MHS access the directory to resolve the corresponding O/R address. It is expected that directory names will be the preferred method of

identifying MHS users as directories become more prevalent, but not all users or DLs will be registered in a directory.

An O/R address contains a set of standard attribute lists that enables the MHS to uniquely identify a user and to deliver a message or return a notification. There are various forms of O/R addresses that can be used for different purposes[4]:

- Mnemonic O/R address provides a means of identifying MHS users or DLs (in the absence of a directory).
- Terminal O/R address provides a means of identifying users with network addresses and/or terminal identifiers.
- Numeric O/R address provides a means of identifying users by use of a numeric identifier.
- Postal O/R address provides a means of identifying originators and recipients of physical messages.

The mnemonic O/R address would be the most prevalent address form as it maps directly to most of the current electronic mail users. Table 6 list the attributes required for such an O/R address.

Table 6: Mnemonic O/R Address attributes

Attribute Type	Description
Personal-name	This value comprises the following pieces of information, the first mandatory, the others optional: • surname, • given-name, • initials (except surname), • generation.
Common-name	Identifies a user relative to another attribute. (eg Marketing Manager.)
Organization-name	Identifies the organisation relative to the country or the administration or private management domains.
Organizational-unit-names	Identifies one or more units (divisions or departments) of the organisation. Each value in the sequence being subordinate to the previous.
Domain-defined	One or more attributes whose type is bound to a class of information by the user's management domain.
Private-domain-name	Identifies the PRMD
Administration-domain-name	Identifies the ADMD. The value of a single space (if permitted by the country) denotes all ADMDs in the country.
Country-name	Identifies the country either by a X.121 numeric string or an ISO 3166 character pair.

Only the Administration Domain Name and the Country Name are mandatory attributes. The rest are optional, but the more attribute values supplied the greater the chance of successful

4. A new O/R address form has also been proposed that will support Internet format addresses to enable interworking from MHS to the Internet (ISO/IEC JTC 1/SC 18/WG 4 N 2556, 26 Oct 1993)

delivery. There is also the flexibility to include Domain-Defined (DD) attributes that allow for previously agreed types to be included in an O/R address.

MHS use of directory

The directory provides capabilities that are beneficial for users in the operation and management of MHS and provision of services. These directory capabilities fall into the following four categories:

1 User-friendly naming. The originator and recipient of a message can be identified by means of the directory name, rather than the machine oriented O/R address.
2 Distribution lists. An originator may supply the name of the DL and the MHS can obtain the directory names (and then the O/R addresses) of the individual members.
3 UA capabilities. Capabilities of a recipient (or originator) user agent can be stored in the directory entry and used to determine MHS preferences and acceptable message body types.
4 Authentication. Two MHS functional entities (MTAs and UAs) can each establish the identity and authentication of the other before communication.

Distribution lists

The ability to make use of a distribution list (DL) is an optional capability of MHS and allows a sender to have a message transmitted to a group of recipients. The group name is used as the message recipient instead of having to specify each of the individual recipients.

The DL can be described as having the following attributes (which may be stored in a directory):

- DL members. Users (including other DLs) that will receive messages addressed to the DL.
- DL submission permissions. A list of users and other DLs which are allowed to send messages to the DL.
- DL expansion point. The DL's O/R address identifies the expansion point (MTA) where the names of the members of the DL are added to the recipient list.
- DL owner. A user who is responsible for the management of a DL.

Submission of messages to a DL is similar to the submission of a message to a user. The MTA responsible for expanding the DL will verify the submission permissions and add the members of the DL to the message recipient list. Since a member of a DL may be another DL, the MTS keeps a record of the expansion history to prevent recursive expansion from occurring.

3.3.4 Interpersonal Messaging System

The interpersonal message service (IPM service) provides a user with features to assist in communicating with other IPM service users. It is intended for (human) users in the exchange of typical electronic mail messages with the added features of the IPM service capabilities and utilising the MT service for sending and receiving interpersonal messages.

The UAs used in the IPM service (IPM UAs) comprise a specific class of cooperating UAs with particular features and functionalities. The optional access units allow for teletex (TLMA), telex (PTLXAU), and physical delivery (PDAU) users to intercommunicate with the IPM service users. The message store can optionally be used by IPM users to take delivery of messages on their behalf and then used by the IPM UA.

The interpersonal (IP) class of UAs create messages containing a content specific to the IPM that is sent from one IPM UA to another. This message type is called an IP-message. The structure of an IP-message, contrasted to a typical office memorandum, is shown in Figure 28. The IP-message is conveyed as the content (with an envelope) when being transferred through the MTS.

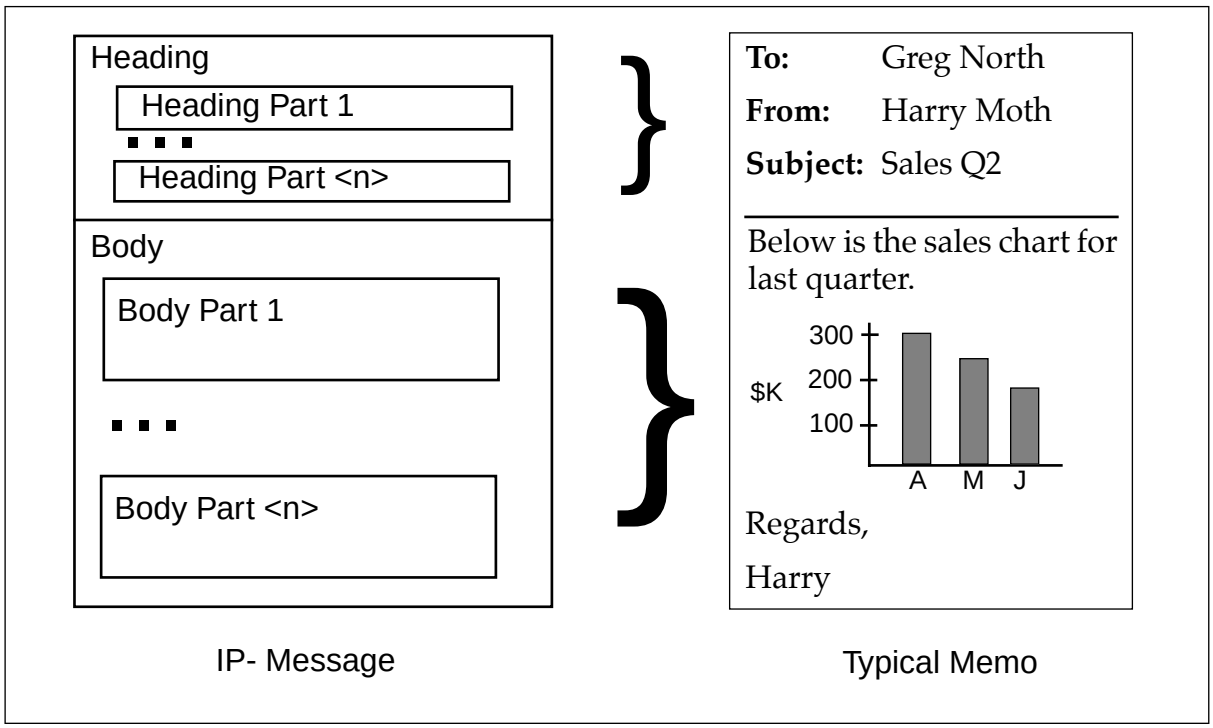


Figure 28: IPM Message Structure

The IP-message contains information (eg to, subject, date) provided by the user which is transformed by the IPM UA into the heading of the IP-message. The heading is made up of many heading fields. The main information that the user wishes to communicate (the body of the memo) is contained within the body of the IP-message. In the example shown, the body contains three instances of two types of encoded information; text, graphic, and text. The IP-message body part can consist of a number of body parts, each which can be of a different encoded information type, such as text, voice, facsimile and graphics.

The IP-message contains a number of heading fields that convey the recipients of the message plus a number of other options that the user may request. The understanding and interpretation of the IPM heading fields are crucial to the successful functionality of the UA. They reflect the possible options for an IP-message and carry status information for the elucidation by the end recipients. A lack of understanding or misinterpretation by the message originator can cause the message recipients to similarly portray the original message content. The heading fields rely on component types of information which are used in the specification of a number of heading fields. These component types include the IPM Identifier, Recipient Specifier, and O/R Descriptor and are fully outlined in Table 7. Aspects of the component types are necessary for the complete appreciation of the functionality provided by the heading fields. Various parts of the component types and heading fields are also classified as mandatory (M), conditional (C), or optional (O).

Table 7: IPM heading field component types

Component Type	Component Description
IPM Identifier	Unambiguously and uniquely identifies an IPM and has the following components: • User (O)—O/R address of the originator. • User-relative-identifier (M)—unique integer or string of characters.

Recipient Specifier	Identifies the (preferred) recipient of the IPM and has the following components:
	• Recipient (M)—an O/R descriptor.
	• Notification Requests—includes one or more of the following (default no values):
	  • non-receipt notification (nrn),
	  • receipt notification (rn) —can only be selected if rn is also selected,
	  • ipm-return—the IPM body to be returned in any nrn.
	• Reply-requested—boolean request for a reply from the recipient (default false).

O/R Descriptor	Identifies a user or DL (an O/R Name) and has the following components:
	• Formal-name (C)— identifies the user. Component is conditional if one or more of the following are satisfied:
	  • The free-form-name component is absent
	  • the O/R descriptor appears in the Reply-Recipients heading field
	  • The O/R descriptor is the Recipient component of a recipient specifier.
	• Free-form-name (O)—string identifying the user.
	• Telephone-number (O).

The sixteen IPM heading fields are representative of the typical requirements for message handling systems. These include four types of recipients and specification of a related user who has authorised the message. A number of heading fields capture relationships with other existing messages as well as time related options. Two heading fields allow the user to attach importance and sensitivity attributes to a message. These are not interpreted by the MHS and can only be given meaning by human users. A supplementary 'extensions' heading field is also included to cover requirements not covered by the existing set of fields. The fields that appear in the IPM heading are described in Table 8.

Table 8: IPM Heading Fields

Heading Field	Heading Field Description
This-IPM	This field comprises an IPM identifier (M)
Originator	Comprises the O/R descriptor (O)
Authorizing Users	Zero or more O/R descriptors for authorising users (C)
Primary Recipients	Identifies zero or more users or DLs who are the primary recipients of this IPM (comprises a sequence of Recipient Specifiers).

Table 8: IPM Heading Fields *(continued)*

Heading Field	Heading Field Description
Copy Recipients	Identifies zero or more users or DLs who are the copy recipients of this IPM (comprises a sequence of Recipient Specifiers).
Blind Copy Recipients	Identifies zero or more users or DLs who are the blind copy recipients of this IPM (comprises a sequence of Recipient Specifiers). This conditional field shall be absent (or empty) in the instance that the IPM is intended for a primary or copy recipient.
Replied-to IPM	This IPM-Identifier field shall be present if the IPM is a reply (C).
Obsoleted IPMs	Zero or more IPM Identifiers that the authorising user(s) of the present IPM consider to be obsolete.
Related IPMs	Zero or more IPM Identifiers that the authorising user(s) of the present IPM consider to be related to it.
Subject	Identifies the Subject of the IPM as a printable string (O).
Expiry Time	Comprises the date and time after which the authorising user(s) consider the IPM to lose its validity (O).
Reply Time	Comprises the date and time that the authorising user(s) requests any reply be sent to the present IPM (O).
Reply Recipients	Identifies zero or more users or DLs who are requested to be among the preferred recipients of any replies to this IPM (C). It comprises a sequence of O/R descriptors. This conditional field shall be present if the originator is not the only specified reply recipient.
Importance	Identifies the importance of the IPM which may be one of: • low, • normal (default), or • high.
Sensitivity	Identifies the sensitivity of the IPM (C) which may be one of: • Personal, • Private, or • Company-Confidential. This conditional field shall be present only if the IPM has been identified as being sensitive.
Auto-fowarded	Indicates whether the IPM was auto-forwarded (default is false).
Extensions	Conveys information accommodated by no other heading field. Comprises of a set of heading extensions with type and value components.

An IPM contains any number and type of body parts. The body consists of a sequence of body-parts that are prescribed by a set of encoded information types (EITs). The IPM UA is

required to provide the functionality to create such multitype body parts. The following EITs have been identified in the X.400 standard:

- IA5 Text
- G3 Facsimile
- Teletex
- Encrypted
- Mixed-Mode
- Nationally-Defined
- Voice
- G4 Class 1 Facsimile
- Videotex
- Message (IPM)
- Bilaterally-Defined
- Externally-Defined

Not all the EITs have been fully defined and may come from other non-MHS standards. The IPM UA must support these EITs for the IPM to be successful for both the origination and reception of messages. Body parts can be converted, if necessary, to be successfully delivered to a class of IPM UA. In some cases, non-delivery may occur if the IPM UA is unable to convert the body part and/or the recipient has specified that the message is not to be converted.

IPM Notifications

In the IPM service a user can request a notification (IPN) of receipt or non-receipt of a message by a recipient. These notifications are requested by an originator and are generated as a result of some recipient action (such as reading a message) or generated automatically by the recipient's UA (in the case of a non-receipt notification).

An IPN may take either of the following forms:

- non-receipt notification (NRN)
- receipt notification (RN)

The IPN comprises a set of common fields and either a set of non-receipt fields (for NRNs) or receipt fields (for RNs). The common fields include the subject (the original IPM Identifier), the IPN originator, the preferred recipient of the original IPM (if required) and any conversion EIT descriptors.

A NRN is generated if the message could not be delivered for a number of reasons including discarded, obsoleted, or time expired. The non-receipt fields are defined and described in Table 9.

Table 9: IPN Non-receipt fields

Field Name	Description
Non-receipt Reason	Indicates why the IPM was not received (M). May be any one of the following values: • ipm-discarded. • ipm-auto-forwarded.
Discard Reason	Indicates why the IPM was discarded (C). May be any one of the following: • ipm-expired—time expired IPM. • ipm-obsoleted —IPM was identified as being obsolete. • user-subscription-terminated—the IPM subscription of the NRNs originator was terminated. This conditional field shall be present only if the Non-receipt Reason field has the value ipm-discarded.

Table 9: IPN Non-receipt fields *(continued)*

Field Name	Description
Auto-forward Comment	This conditional field shall be present only if the Non-receipt Reason field has the value ipm-auto-forwarded and the auto-forward comment string was supplied.
Returned IPM	This field shall be present only if the ipm-return is among the values of the Notification-requests components and consists of the IPM (C).

A RN is generated when the message is received at the recipients UA and indicates the time the IPM was read by the intended recipient and the acknowledgement mode. The receipt fields are defined and described in Table 10.

Table 10: IPN Receipt Fields

Field Name	Description
Receipt Time	Date and time when the IPM was received(M).
Acknowledgement Mode	Identifies the manner in which the RN was generated: • manual —the RN was originated by the actual recipient of the IPM. (The default value) • automatic—the RN was originated as a result of auto-acknowledgement being set
Suppl Receipt Info	Printable string giving supplementary information about the receipt of the IPM (O).

3.3.5 MHS Protocols

The communications between the MTA, UA and MS are encapsulated by a number of protocols:

- P1—MTS Transfer Protocol between two MTAs.
- P2—IPM Protocol.
- P3—MTS Access Protocol between the UA or MS and the MTA.
- P7—MS Access Protocol between the UA and the MS.

The X.400 system totally resides in the OSI Application Layer with the MHS protocols utilising distinct Application Service Elements (ASEs). These include common service elements such as Association Control Service Element (ACSE), Remote Operations Service Element (ROSE), and Reliable Transfer Service Element (RTSE).

The P1 protocol supports the MTA to MTA communications conveyed as Application Protocol Data Units (APDUs) by the RTSE. The P1 protocol consists of the following service operations:

- Bind and Unbind,
- Message (user-generated message),
- Probe (to test the deliverability of a message), and
- Report (delivery and non-delivery notifications).

The P2 protocol is for IPM contents that are delivered between UAs that are defined by the Interpersonal Messaging System. The information (IPM or IPN) that is carried is encapsulated inside the P1 or P3 protocol.

The P3 protocol supports the submission and delivery between the UA or MS and the cooperating MTA and is conveyed as a set of Remote Operations using the ROSE protocol. The P3 protocol consists of the following service operations:

- Bind and Unbind
- Probe submission
- Submission control
- Message delivery
- Register
- Message submission
- Cancel deferred delivery
- Delivery control
- Report delivery
- Change credentials

The P7 protocol supports the message interrogation and management between the UA and the MS and is implemented utilising Remote Operations (using ROSE). The P7 protocol consists of the following service operations:

- Register
- List
- Delete
- Summarise
- Fetch
- Alert

3.3.6 MHS Service Elements

Elements of service are particular features, functions, or capabilities of the MHS and are associated with the various services provided in MHS. These service elements can be classified into the following:

- MT service capabilities for sending and receiving messages between UAs.
- PD service for MH users to send messages and have them delivered in a physical medium to non-MH users.
- MS elements of service specifically available for the use of message stores.
- IPM service capabilities which provide for the sending and receiving of messages between IPM UAs.

The elements of service are further classified as:

1 basic, or
2 optional.

The elements of service belonging to a basic service are inherently part of that service and should always be provided to the users of that service. Optional elements of service can be selected by the user either on a per-message basis or for an agreed period of time. Each optional element is also classified as either essential (and available to all MHS users) or additional (made available on the basis of bilateral agreements).

The basic MT service elements enables a UA to submit unique and unambiguously identified messages and to have such messages delivered to it. Non-delivery notifications can be sent to the originating UA if the message was unable to be delivered. UAs can also indicate the encoded information types (EITs) that it can process to facilitate efficient communications. Messages will also summarise the EIT parts and an indication of any conversions that have been performed (and the resulting EITs). The submission time and delivery time of the message is

also supported. The complete basic and optional MT service elements are listed in Appendix A.

The basic PD service enables messages to be delivered to recipients in a physical (hard copy) format via a physical delivery service such as the postal service. The optional PD user facilities can be used together for the provision of an enhanced MT service. The complete basic and optional PD service elements are listed in Appendix B.

The basic MS service elements are available to provide for storage and management of messages acting as an intermediary between a UA and an MTA. The optional MS user facilities can be used together for enhanced use of a message store. The complete basic and optional MS service elements are listed in Appendix C.

IPM Basic Service Elements

In the basic IPM service, a user prepares and sends an IP-message, which makes use of the MT service, to facilitate communication between respective users. The user specifies the O/R name of the recipients who are to receive the uniquely identified IP-message. Following a successful delivery to the recipient's UA, the IP-message can be received and acted on by the recipient. The basic IPM service elements dictates the minimum functionality of a IPM UA and hence, has great importance on the user interface. Each element of service requires some form of user interaction, from a simple status flag to a complex interaction dialogue. The IPM elements of service belonging to the basic IPM service are listed in Table 11.

Table 11: IPM Basic Service Elements

Elements of Service
Access management
Content type indication
Converted indication
Delivery time stamp indication
IP-message identification
Message identification
Non-delivery notification
Original encoded information types indication
Submission time stamp indication
Typed body
User/UA capabilities registration

Local IPM UA facilities can usefully provide the optional user facilities of the IPM service, which can be selected on a per-message basis or for an agreed period of time. Additionally, the optional IPM service elements are classified for both origination and reception by UAs. If the Management Domain supports the optional service elements, then originators of IP-messages can utilise the procedures defined for the associated element of service. If the MD offers these optional service elements for reception, then the receiving UA will be able to re-

ceive and recognize the element of service and inform the user of the requested optional user facility.

The IPM optional service elements classifies the Origination and Reception elements as either additional or essential. These are listed in Appendix D including optional IPM service elements that can be selected for a contractual period of time. With so many IPM service elements available, it is important that an appreciation for the complexity of these facilities offered by the IPM system is attained. The corresponding implications for the design of IPM UAs is also of foremost concern.

3.3.7 Research and Applications of X.400

The X.400 MHS has spawned extensive research into the application of the standard to network environments and value-added extensions to facilitate other MH services. One such use is for Electronic Data Interchange (EDI) for the transmission of business related documents between organisations. X.400 seems a likely candidate as the MTS as a new X.400 series standard is proposed to support EDI (Genilloud, 1990). Other work includes development of multimedia MHS based systems (Ohmura *et al*, 1986), object–oriented approaches to the functional X.400 view (Lung & Swee, 1991), and X.400 protocol test systems such as GENEPX400 (Goyer & Radureau, 1988). Tse *et al* (1988) also reports on the results of testing two separate X.400 implementations and the issues involved in their interworking across country boundaries. The X.400 standard is also being considered as the basis for the support of asynchronous group communication tasks such as computer conferencing and bulletin board applications (Palme, 1993).

Other research has made critical analysis of the standard and identifies some areas in which misinterpretations of the architectural concepts and 'overdesign' of the MTS may impair the implementations (Sinderen & Dorregeest, 1988). Reports of other studies indicate that the complex addressing, limited support for network transports[5], and costs are prohibiting X.400 growth (Gillooly, 1991).

X.400 has also been the subject of research into the advantages that such an enterprise-wide message system will have on an organisations efficiency and market strength. Houldsworth (1990) and Katsaros (1990) conclude that X.400 offers many business advantages including the ability to scale as the organisation grows. Whitten (1989) also concludes that X.400 offers strategic advantages that will affect businesses long-term planning and change the way that they communicate. Walravens (1991) reports that the market for X.400 and electronic mail value added services will reach ECU1,516 million with an annual growth rate of 65% in Europe alone. EDI products based on X.400 are also continuing to influence business decisions on messaging systems such as (Tandem Computers, 1992).

General reviews of the X.400 standard can be found in (Vervest, 1987; Law, 1989; Schicker, 1989; Chilton, 1990; Radicati, 1992a). Examinations of the X.400 standard with respect to the OSI Application Layer can be found in Chapter 11 of (Bijendra & Agrawala, 1990), Chapter 10 of (Henshall & Shaw, 1990), Chapter 11 of (Black, 1991), and Chapter 15 of (Dickson & Lloyd, 1992).

---

5. Initial implementations of X.400 were only available on X.25 and OSI networks. Recent implementations run over TCP/IP networks.

Numerous X.400 implementations are currently available with many still being based on the 1984 X.400 standard. Scott (1990), King, Julia (1992), and Burns & Radicati (1993) contrast and review many of these implementations and report on their interoperability with other X.400 systems and gateways. Others, such as (Mierswa, 1989) report on X.400 implementations that are used by proprietary systems to also connect to other vendors implementations. Laborie & Huitema (1988) report on the interconnection of their X.400 system to various networks and the incompatibilities found in the protocols and the interpretations of the standard.

Development of X.400 gateways into international notable networks (Magedanz *et al*, 1988) and with other national networks (Henken, 1988) has seen the dispersion and realisation of X.400. Gateways such as Connect400[6] allow organisations to extend the scope of their communications by allowing existing electronic mail systems to transfer mail to other disparate networks. Implementations for distributed X.400 UAs (Greene & Tissot, 1988) and mobile access architectures for X.400 services (Cole & Burns, 1989) have also been developed.

X.400 has also been the subject of analysis to determine the mappings between MIME based messages and X.400 message formats. In particular, the mappings between the various IPM body parts and the corresponding MIME body parts (Blum, 1992). With the popularity of MIME increasing, the results of this research may influence an organisation's electronic mail standard.

A comprehensive list of X.400 implementations can be found in (Alvestrand, 1993) and a current review of X.400 integration servers in (Blum & Rowe, 1993). An overview of the PP X.400 implementation is described in section '4.5 PP X.400 MHS Overview' on page 103.

With X.400 being such a complicated and large standard, APIs have been developed to provide a stable framework for the implementations of X.400 systems. Miller *et al* (1990) and Koorland (1991) discuss the 'X.400 Application API' and the 'X.400 Gateway API' which are both based on the OSI Object Management API (Omnicom, 1990). (The Object Management API provides an object-oriented common workspace for OSI application-specific APIs.) Both of these APIs provide conformance to the standard and a strategic architecture to build messaging applications and gateways. Other APIs include Apple Computer's MacX.400 product (Salamone, 1992). Apart from APIs, CCITT offers a guide to help implementors of X.400 systems by outlining and discussing the defects and reporting corrections to the standard (CCITT, 1992).

The security aspects of X.400 are one of the key features of the standard. The success of the security features will enable X.400 to enter the security-conscious financial markets. Some organisations have developed systems with extra security features (far beyond those defined in the standard) which meet the US Department of Defences highest security levels (Mantelman, 1989).

Other research (Mitchell *et al*, 1989; Mitchell *et al*, 1990; Mitchell, 1990) has highlighted limitations to the security aspects of X.400 including the following:

- Proof of delivery to a UA is not available when an MS is used.
- Proof of delivery by an MS is not possible if the message content is encrypted.
- Computation of the crypotographic tokens signature may lead to problems.

---

6. Connect400 is a registered trademark of Network Innovations Pty Limited.

X.400 Mapping to RFC 822

RFC 1148bis (Hardcastle-Kille, 1991a) defines a set of mappings which enable interworking between X.400 MHS and systems using RFC 822 mail protocols. The mappings aim to maximise the services offered across the boundaries and does not require any changes to the end systems. RFC 1148bis covers the mappings required for the service elements, character sets, protocols elements, and addressing. Of course, some elements of IPMs cannot be supported by RFC 822 (for example, body parts with non-text EITs and Alternate Recipient Allowed service elements). Allocchio & Ghiselli (1990) describe the GIVEME 987 gateway an report on its success with one X.400 proprietary system. The main problems of coordinating X.400 to RFC 822 mapping gateways are not usually technical but depend on national and international bodies agreeing on a standard to adopt across all gateways (Hansen, 1990).

As an example of address mapping, consider an RFC 822 user wishing to send a message to the O/R Address:

C=AU; ADMD=Telememo; PRMD=OZ; O=Bond; S=Smith; G=Fred;

The gateway and mapping table lookups would form an RFC 822 address:

Fred.Smith@bond.edu.au

For the opposite case, consider an X.400 user wishing to send a message to the RFC 822 address:

mary@bond.edu.au

The X.400 user would use the following address:

C=AU; ADMD= ; PRMD=Internet; DD.RFC-822=mary(a)bond.edu.au

In this case the Domain-Defined attributes are used to encapsulate the RFC 822 part of the address (Hansen, 1991). Earlier mappings attempted to match each subdomain of the RFC 822 address with an attribute of the O/R address (Henken, 1987; Ullmann, 1989) but have proven unable to cope will all cases of RFC 822 addresses.

3.4 X.500 Directory Services

In 1988, CCITT released the X.500 series recommendations on Directory Services to store information on objects (such as people, devices, and applications) and used to facilitate communication between such objects. The directory provides services within the following environments:

- Telecommunications networks that will constantly undergo change in the characteristics of the network objects.
- The lifetime of objects is not short and will be involved in communications more frequently than changes in address, location etc.
- Objects are typically identified by numbers or strings of symbols that are selected for their ease of allocation and not for ease of use for humans.

One of the key motivations for the directory was to allow for user friendly names for X.400 message addressing. The directory can be used to 'look up' a message recipient's X.400 address given such information as the recipients name and organisation (see ' MHS use of directory' on page 63). The directory is used to handle the name-to-address mapping.

The full X.500 series of recommendations (CCITT, 1988b) is listed in Table 5 with the corresponding ISO standards.

Table 12: X.500 Directory Recommendation Series

CCITT	Name of Recommendation	ISO
X.500	The Directory—Overview of Concepts, Models and Services	9594–1
X.501	The Directory—Models	9594–2
X.509	The Directory—Authentication framework	9594–8
X.511	The Directory—Abstract Service Definition	9594–3
X.518	The Directory—Procedures for Distributed Operations	9594–4
X.519	The Directory—Protocol Specification	9594–5
X.520	The Directory—Selected Attributes Types	9594–6
X.521	The Directory—Selected Object Classes	9594–7

The directory defines rules for naming objects that can be accessed by other application entities or persons to obtain information about the objects. The types of objects are arbitrary although X.521 defines a standard set of objects. Externally defined ‘classes’ of objects are also supported.

3.4.1 The Directory Model

The X.500 Directory Services is logically a very large database of objects. Physically, X.500 is setup as a distributed database with cooperating agents providing the directory functions.The directory is based on the client-server paradigm with Directory User Agents (DUAs) interacting with Directory System Agents (DSAs). The DUA is the access method used by humans or application entities to interrogate the directory (a series of cooperating DSAs) holding the database information. The DSAs have the ability to pass on operations to other DSAs if the information is not locally available. Figure 29 shows the DUA and DSA interactions of the X.500 Directory Services model.

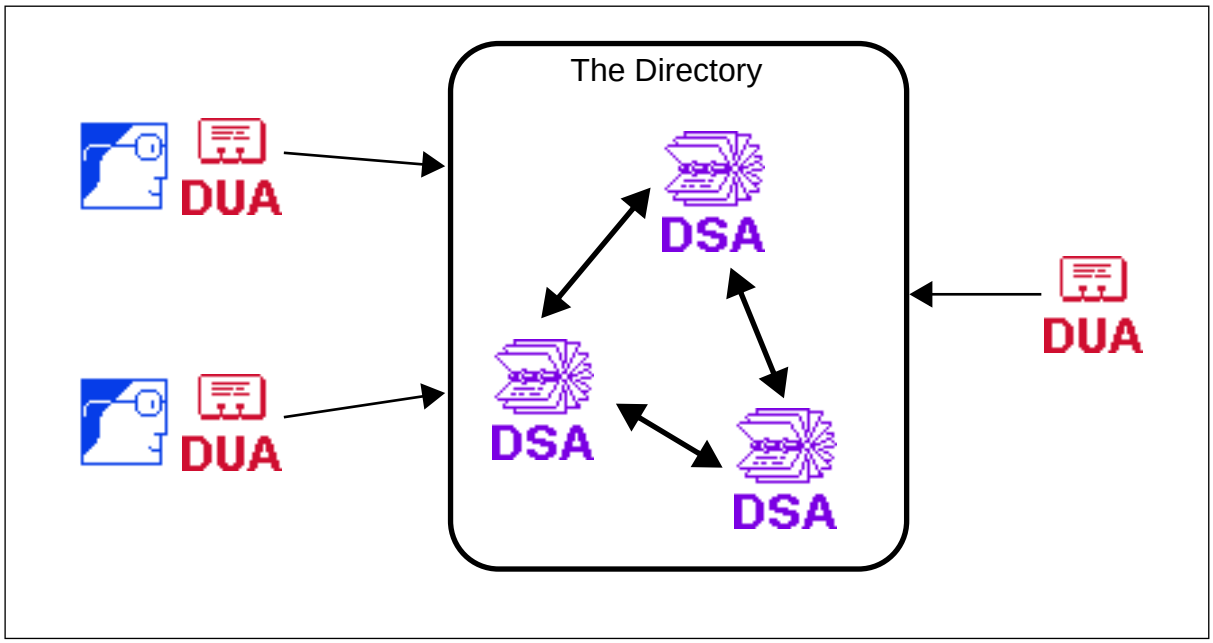


Figure 29: The X.500 Directory Services Model

The X.500 directory specifies:

- functions of the DUA,
- operations of the DSA,
- directory protocols, and
- representation of information for objects.

Two key areas that the directory standard does not specify is the user interface for the DUA or the method of database storage for each DSA.

The distributed database nature of the Directory provides the following benefits:

- manageable size (each organisation maintains their own data in the DSA), and
- fault tolerance.

3.4.2 Directory Information Base and Tree

The complete directory information is called the Directory Information Base (DIB) and consists of entries for each object in the database. An object's entry contains a set of attribute-value pairs. Each attribute has a type (eg string, integer) and zero to many values. In the case of an entry having multiple values for an attribute, the first value is taken as the distinguished value.

The DIB is arranged in an hierarchical Directory Information Tree (DIT). Each entry in the DIB is either an arc or leaf node in the DIT and is based on the entry's relationship with its parent node. The DIT has a root (logical) entry at the top of the tree. Figure 30 shows the DIT structure and the attribute-value scheme used for entries in the DIT.

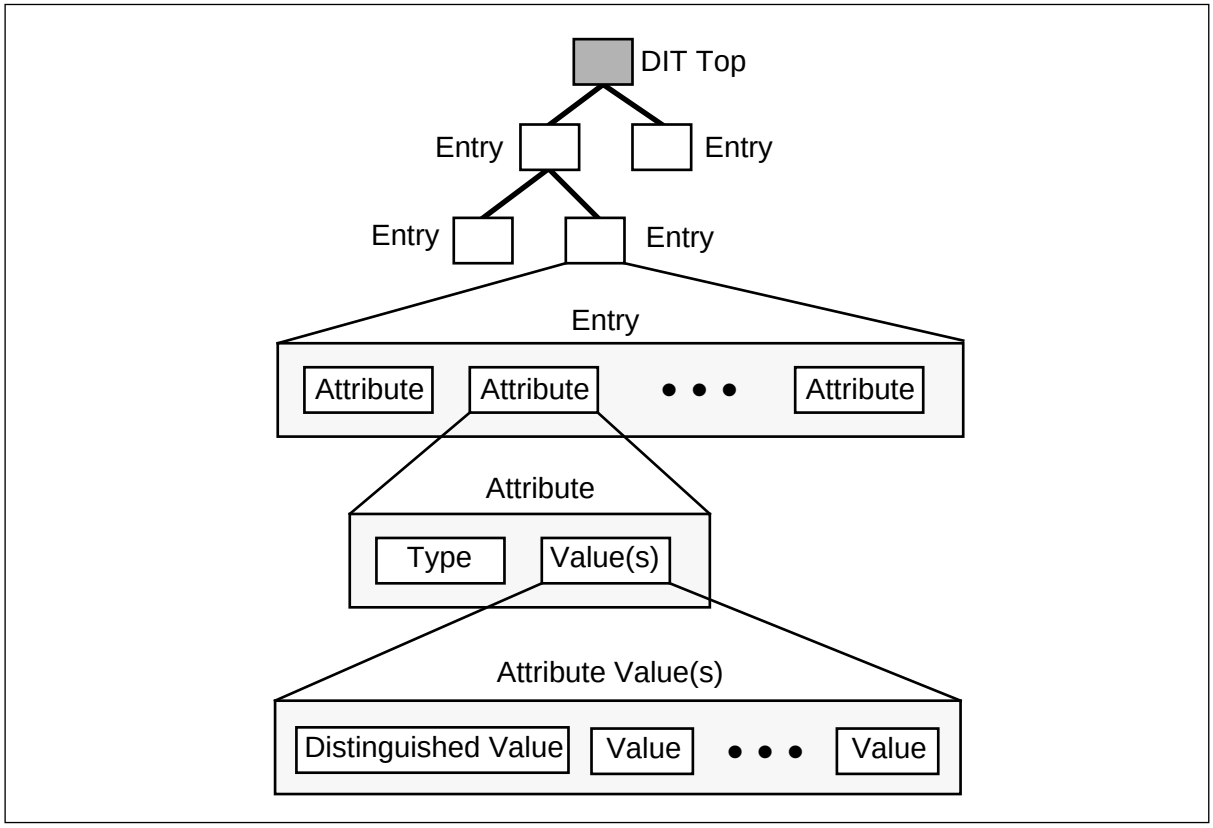


Figure 30: DIT Entry Structure

3.4.3 Directory Object Classes

The directory defines a number of classes of objects that may be useful for applications of the directory. Each entry must be a member of at least one object class. The object class defines the set of attributes that may be used in the object's entry in the DIB. The class definitions also allow for others classes to be derived from the standard set to generate new object classes (inheriting all the base class attributes). Table 13 list the standard object classes defined in the directory.

Table 13: The Directory Object Classes

Object Class Name	Object Class Name
Top	Group of Names
Alias	Residential Person
Country	Application Process
Locality	Application Entity
Organization	DSA
Organizational Unit	Device
Person	Strong Authentication User
Organizational Person	Certification Authority (CA)
Organizational Role	

One of the advantages of having a series of standard classes is that some of the attributes may also be inherited to a derived object's entry. In the case where no value for an attribute is specified, then the parent classes' value for that attribute may be used. For example, the facsimile number attribute for an Organizational Unit object may be inherited by subordinate Organizational Person objects. In this case, if the facsimile number changes, then only one entry in the DIT is changed for all the corresponding derived objects.

3.4.4 Directory Attribute Types

The directory defines a number of standard attributes that represent information that a class of objects may require. Each entry (being a member of a class) may use the attribute defined for that class. The directory allows new attributes to be defined but these may not be available to other users beyond the authority which created them.

Table 14 lists the standard set of attributes grouped into general types.

Table 14: The Directory Standard Attributes

Attribute Group	Attribute Names
System	Object Class, Aliased Object Name, Knowledge Information
Labelling	Common Name, Surname, Serial Number
Geographical	Country Name, Locality Name, State or Province Name, Street Address
Organizational	Organization Name, Organizational Unit Name, Title

Table 14: The Directory Standard Attributes *(continued)*

Attribute Group	Attribute Names
Explanatory	Description, Search Guide, Business Category
Postal Addressing	Postal Address, Postal Code, Post Office Box, Physical Delivery Office Name
Telecommunications Addressing	Telephone Number, Telex Number, Teletex Terminal Identifier, Facsimile Telephone Number, X.121 Address, International ISDN Number, Registered Address, Destination Indicator
Preference	Preferred Delivery Method
OSI Application	Presentation Address, Supported Application Context
Relational	Member, Owner, Role Occupant, See Also
Security	User Password, User Certificate, CA Certificate, Authority Revocation List, Certificate Revocation List, Cross Certificate Pair

3.4.5 Directory Structure

The directory suggests some common naming practices and DIT structures that may be useful. In general, the classes naming attribute (eg Common Name, Country Name etc) is used as the naming authority. The directory suggests that the DIT follow a pre-defined structure with objects entries added where appropriate. Figure 31 shows an example (using the main object classes from Table 13) of the DIT structure suggested by the X.500 directory. In this example, the arrows indicate the hierarchical relationship between the objects. As can be clearly seen, the structure can take many complex, yet valid, forms.

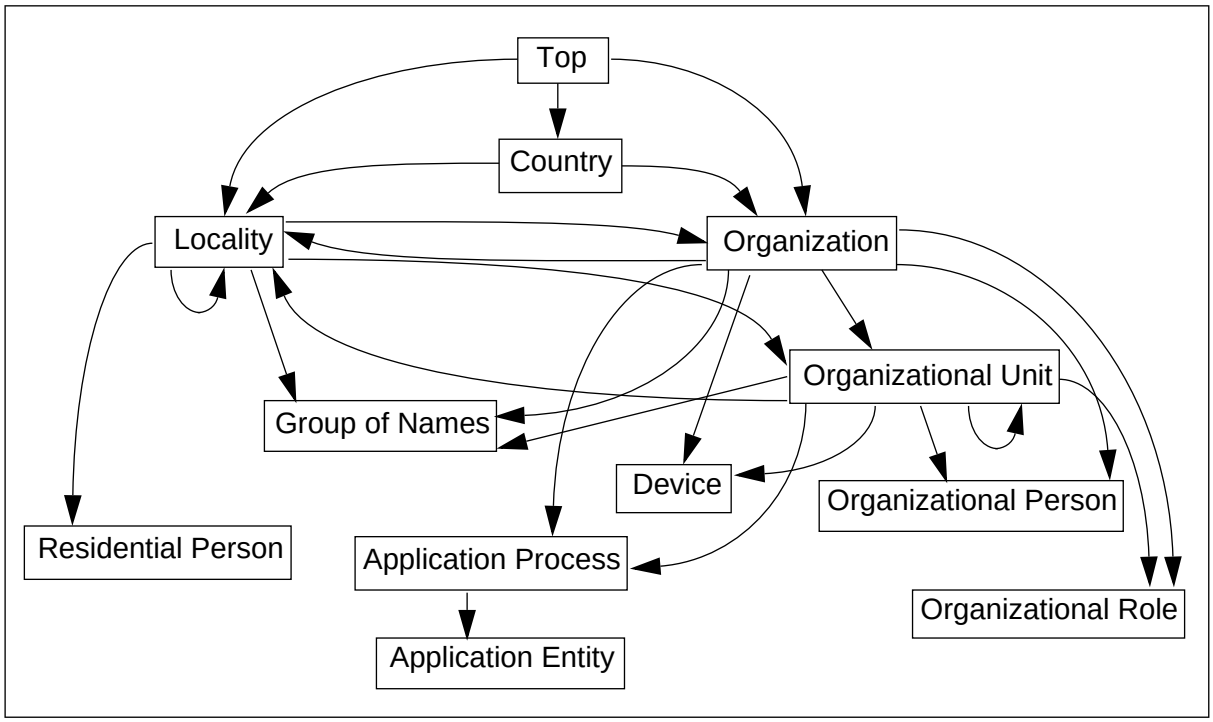


Figure 31: DIT Structure

The X.500 directory schema is a set of definitions and constraints that concern:

- the structure of the DIT,
- naming of entries,
- information in the entries, and
- attributes used to represent that information.

The schema prevents the creation of entries of incorrect object-classes, inappropriate attributes, and invalid attribute values. Formally, the schema comprises a set of definitions and rules which is shown in Figure 32.

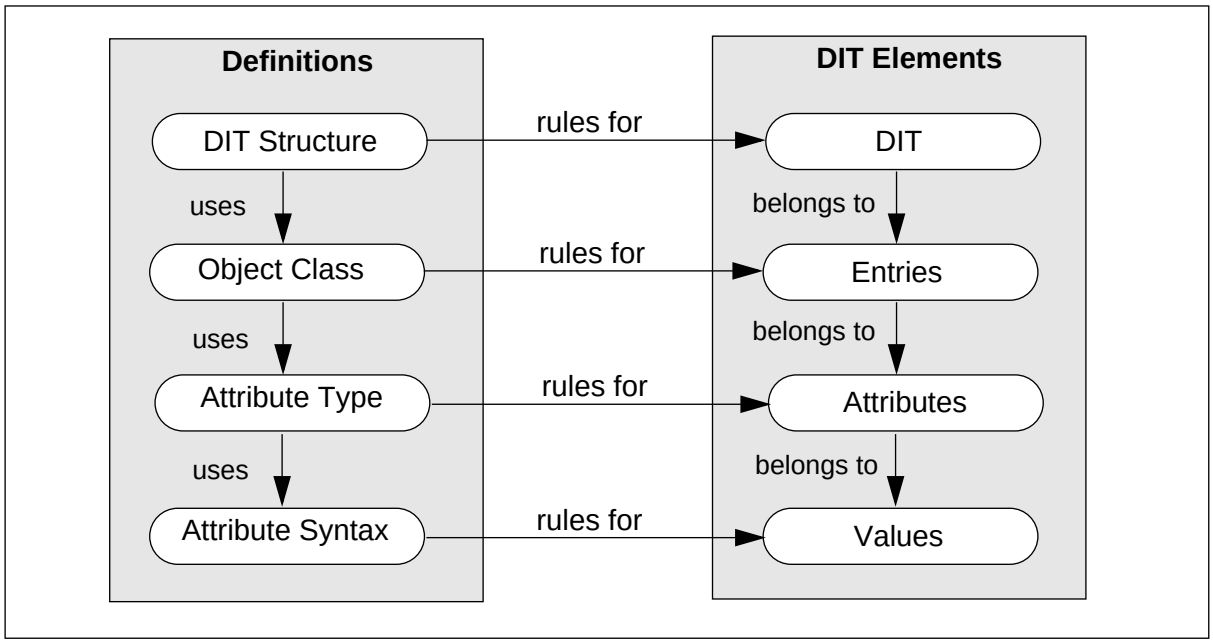


Figure 32: Directory Schema

3.4.6 Directory Names

Each entry has a Distinguished Name (DN) that uniquely and unambiguously identifies the entry and its location in the DIT. The DN is derived from the ordered sequence of the names of the entries from the root of the tree down to the individual entry.

Each entry also has a unique Relative Distinguished Name (RDN) that is a set of nominated attribute values at a particular level in the DIT. The sequence of RDNs from the top of the tree constitute the DN of an entry. Table 15 shows an example of the DN and RDN values for an entry in the DIT.

Table 15: Directory Naming Example

Object Class	RDN	DN
Top		
Country	C=AU	C=AU
Organization	O=Bond University	C=AU, O=Bond University
Organizational Unit	OU=Law	C=AU, O=Bond University, OU=Law
Organizational Person	CN=Fred Smyth	C=AU, O=Bond University, OU=Law, CN=Fred Smyth

Alternative names can be also be used in an entry's RDN to allow for common abbreviations and variations in spelling. For example, the Common Name (CN) attribute for the example shown in Table 15 could also have the alternate names 'Fred Smith' and 'Freddy Smith'. In such cases only one CN attribute value would be used for the DN and RDN of the entry.

Some entries in the DIT may also be alias entries that point to another entry in the DIT and contain no other information. Alias entries provide the basis for multiple names of an object and allows objects to move location in the DIT and still be accessible under previous DNs.

User Friendly Naming

Although not part of the X.500 standard, the directory provides guidelines in naming objects to be as 'user friendly' as possible. In particular, object names should:

- be guessable based on information about the object that humans naturally possess.
- be non-ambiguous (the directory will not assume that a specification identifies one particular object).
- include common abbreviations and variations on spelling (eg Digital Equipment Corporation, Digital, DEC).
- not enforce any precise ordering of multi-part names.
- not artificially remove natural ambiguities (eg two entries with the same surname are allowed in which case the object's initials or first names are used for identification).

3.4.7 Directory Service Operations

The directory supports Read, Search, and Modify service operations between DUAs and DSAs service ports to perform the necessary actions for directory interrogation. The directory also supports Bind (to establish access and exchange credentials) and Unbind operations.

The Read operations include:

- Read (given a DN, returns the set of attributes requested).
- Compare (checks that the attribute/value supplied matches the DN entries—useful for password checking).
- Abandon.

The Search operations include:

- List (return subordinate entries to given DN).
- Search (return specified attributes for all the entries matching some criteria).

Both of these Search operations may return limited results depending on the configuration and administrative policies of the DSAs being searched.

The Modify operations include:

- Add Entry
- Remove Entry (leaf entries only)
- Modify Entry (add, remove, change attributes)
- Modify RDN (specify new set of distinguished values)

Each DSA has *knowledge* of a finite portion of the DIT—usually its children and parent. The DSA holds references to other DSAs in order to pass on operations that it may not be able to respond to.

These DSA references include:

- Internal
- Subordinate DSAs
- Superior DSAs
- Cross References

Depending on the references of the local DSA, the operations will be passed to other DSAs for processing. If the local DSA has no knowledge of the portion of the DIT that the read, search, or modify operation requires, it will pass the operation to another DSA (using its references list). The DSAs use three techniques in passing operations to other DSAs:

1 Chaining—DSA passes on remote operation to another DSA for processing.
2 Multicasting—DSA passes identical remote operation in parallel to other DSAs for concurrent resolution. If DSA is unable to process operation, it returns no-result.
3 Referral—DSA returns a reference to another DSA that may be able to process the operation based on its greater knowledge.

There are a number of reasons why a DSA would return a referral to another DSA instead of chaining the operation itself:

- The DSA may forbid chaining (administrative policy).
- The DSA may be currently uncontactable.
- The referred DSA may have a charging policy.

Figure 33 graphically shows these three distributed operations.

There are a number of DSA operational configuration options that can be implemented that affect these distributed operations:

- prefer chaining (always chain an operation and do not return referrals)
- prohibit chaining (always return a referral)
- local scope (restricted to operations on the local DSA)
- do not dereference alias (return alias entries, not what the alias points to)
- don't use copy (always read the master data, not replicated data)
- priority (low, medium, high)
- time limit (maximum number of seconds that an operation can be performed for)
- size limit (maximum number of objects that may be returned)
- scope of referral (limit referrals to particular management domain or country)

DIT Entries can have discrete access control categories to protect from unauthorised access. Access is granted based on a users previous authentication with the DSA. Each entries attributes may have the following access controls implemented:

- Detect (detect if the protected attribute exists for an entry)
- Compare (compare the value supplied with the protected entries value)
- Read (read the attributes value)
- Modify (change the attributes value)
- Add/Delete (remove or add a new attribute, value, or entry)
- Naming (modify entries RDN and create subordinate entries)

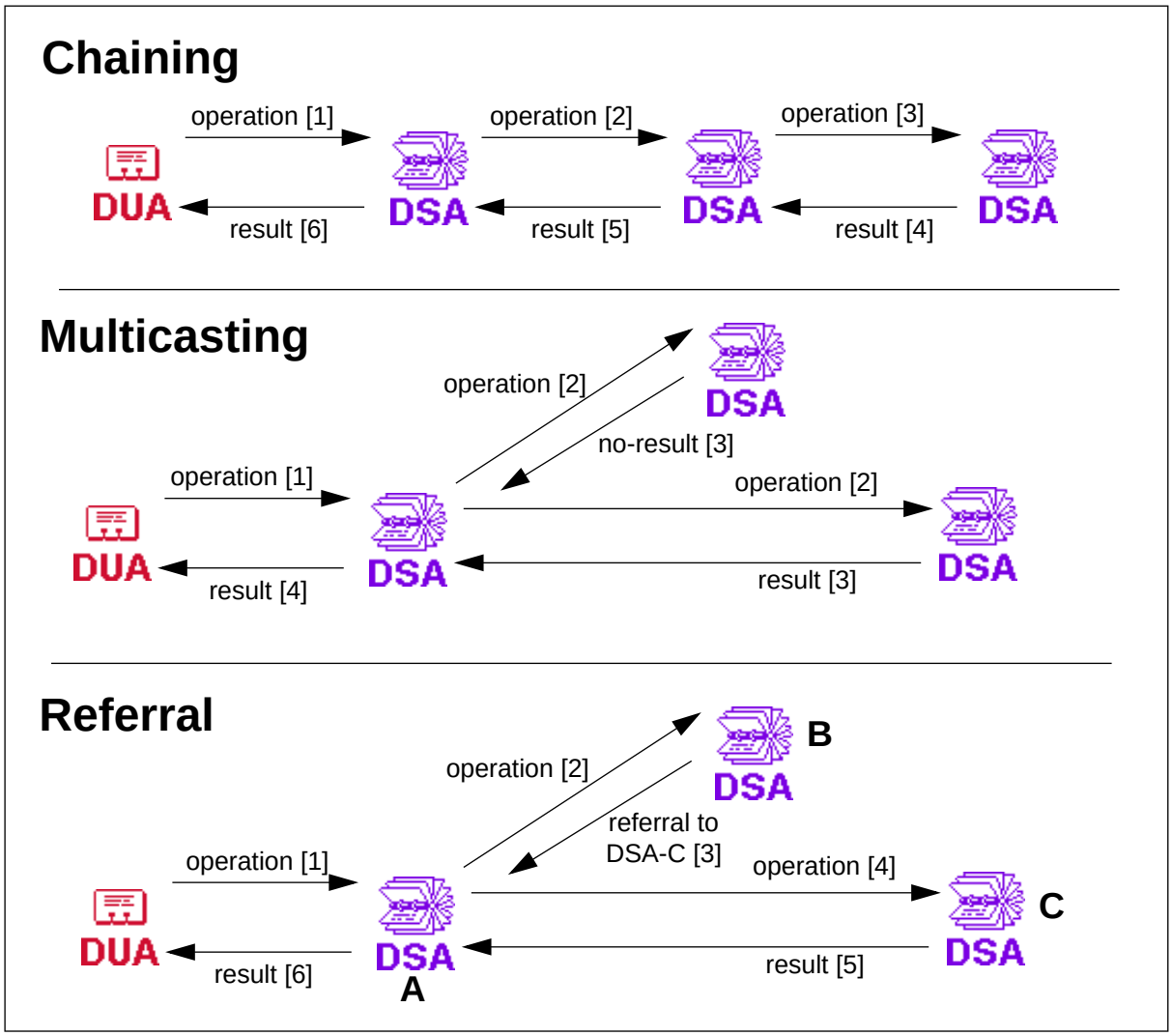


Figure 33: Directory Distributed Operations

The directory allows for *simple* and *strong* authentication. Simple authentication is based on a compare operation with an entry's password attribute. Strong authentication (defined in X.509) provides support for asymmetric public key management schemes. This is the basis of X.400 security services in which the directory stores certified public keys for authentication and encryption.

The Directory supports two OSI Application Layer Protocols:

1 Directory Access Protocol (DAP) between DUA and DSA, and

2 Directory System Protocol (DSP) between DSA and DSA.

DAP supports Read, Search, and Modify Application Service Elements (ASEs) and DSP supports Chained Read, Chained Search, and Chained Modify ASEs. Both protocols utilise underlying Remote Operations (ROSE) and Association Control Service Elements (ACSE). Figure 34 shows a summary of the relationships between the DAP and DSP protocols and the DUA and DSAs.

3.4.8 Research and Applications of X.500

The X.500 standard has generated large amounts of research owing to the standard's flexibility of objects in the DIT and the perceived benefits from a standard for directory systems. Re-

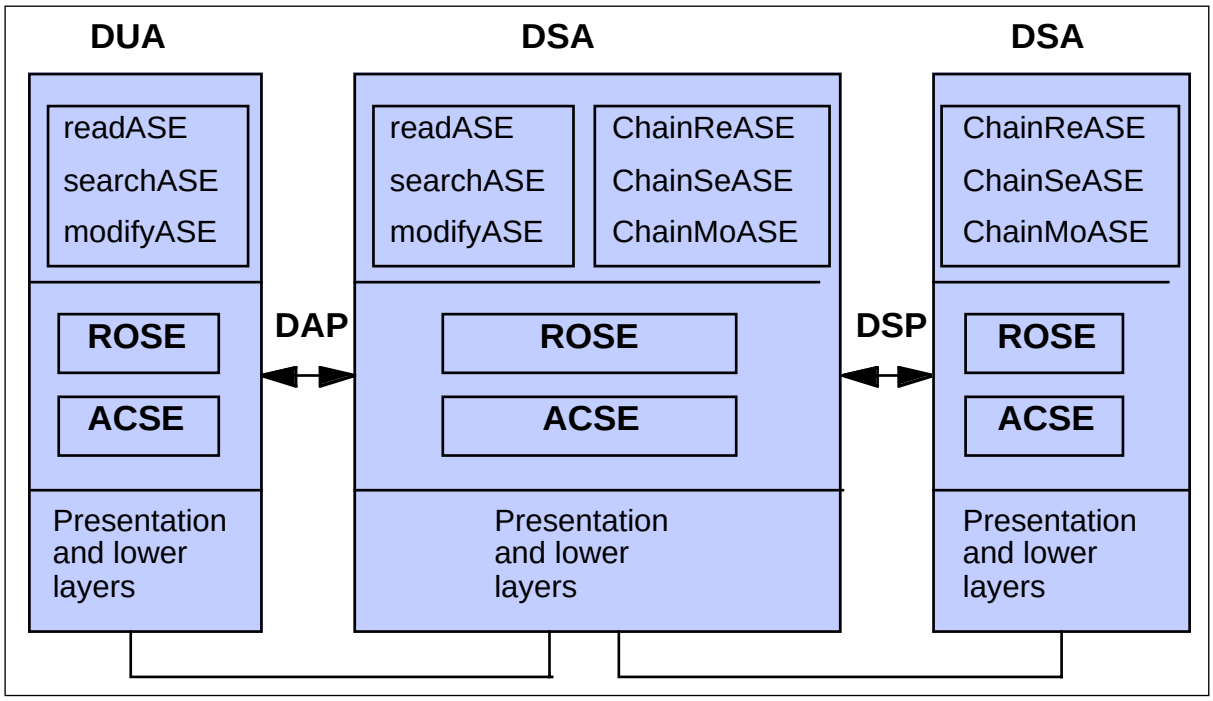


Figure 34: X.500 Protocols

search has covered the full range, from formal specification of X.500 in Z (Wildman & Hayes, 1990), methods of managing an organisation's DIB data derived from multiple sources (Barker, 1991a), and criteria to judge the conformance, performance, interoperability, and metrics of DSAs (Barker & Hardcastle-Kille, 1992b) and DUAs (Barker, 1992). Development of DUAs has been plentiful from the text based *de* (Barker, 1991b) to the graphical *XLookUp* (Mahl, 1991) and *Xdj* (Colville, 1991).

Apart from the standard object classes defined in X.500, others have produced schemas for a broad range of objects. Zahm (1992) defines a schema to be used by hotels and restaurants to enable potential customers to browse the room rates and the 'specials of the day'. The majority of other schemas are all based on networks services. Hong *et al* (1993) describes the potential and feasibility of using X.500 in the network management framework and adapted to a network management monitoring tool. X.500 services have been extended to include use of the Gopher document search protocol (Howes, 1993) and to the File Transfer Protocol (Barker, 1993). A survey of other advanced usages of X.500 can be found in (Weider & Wright, 1993).

X.500 has produced the greatest research interest in the deployment of Internet directory services. These include general resource lookup, advanced user agent, and archive services. Hardcastle-Kille *et al* (1993) describe these aims in detail with proposed solutions to achieve these goals. Other work in this area include:

- Extensions to provide replication and distributed operations on the Internet (Hardcastle-Kille, 1991b).
- A new format for encoding non-OSI network addresses in the directory (Hardcastle-Kille, 1991c).
- Mappings of Internet domains to X.500 (Hardcastle-Kille, 1991d).
- Textual interface to X.500 DAP over TCP/IP connections (Rose, 1991).

Other research has criticised X.500 (Marshak, 1991) and proposed a new two-tier model for naming services with X.500 maintaining the global service connected to high performance lo-

cal servers. May (1992) describes the experiences with the development of X.500 services utilising the XDS API (Omnicom, 1990) which enjoys a large amount of industry support.

General reviews of the X.500 standard can be found in (Payne, 1990; Sykas & Lyberopoulos, 1991; Hunt, 1992). Examinations of the X.500 standard with respect to the OSI Application Layer can be found in Chapter 10 of (Bijendra & Agrawala, 1990), Chapter 11 of (Henshall & Shaw, 1990), Chapter 12 of (Black, 1991), Chapter 16 of (Dickson & Lloyd, 1992) and (Rose, 1992).

There are three significant X.500 pilot projects (Marine, 1993):

1 AARNet Directory Services Project,
2 PARADISE Project, and
3 PSI White Pages.

These are respectively operated from Australia, UK, and the USA and are all interconnected. Other notable X.500 pilot systems are EAN (Neufeld *et al*, 1992) operated from Canada and the Distributed Computing Environment (DCE) directory services (OSF, 1990b). Robbins (1991) discusses how the growth of the directory pilots has lead to a greater structured management of the top-level DSAs in an effort to provide realisation of production directory services. Waugh (1992) also discusses the management of X.500 directory service and the key aspects to operating a DSA.

A comprehensive catalogue of X.500 implementations can be found in (Lang & Wright, 1991). An overview of the QUIPU X.500 implementation is discussed in section '4.6 QUIPU X.500 Directory Services Overview' on page 104.

One of the most important aspects of the directory services is the adoption of a consistent and feasible naming architecture. Barker & Hardcastle-Kille (1992a) and Hardcastle-Kille (1992b) document guidelines for naming organisational and personal entries in the directory pilots including issues relating to the hierarchical model used in the X.500 naming schema. Barker & Kille (1991) specify a complete naming schema for use in the Internet directory pilots. Apart from the set of standard object classes attributes, the schema provides a large number of generally useful object classes and attributes.

One of the goals of X.500 is to achieve user friendly naming with a user-oriented notation and guessability as two main aspects. Hardcastle-Kille (1992a) proposes a standard format for representing names that need to be communicated between users and a matching algorithm. An example of this notation is:

Fred Smith, School of Law, Bond University, AU

Afifi & Huitema (1992) propose a similar system with search algorithms. Both systems do not require the use of attribute types resulting in a less structured free-form notation but requiring complex matching algorithms.

The extensions to the X.500 standard in 1992 focused on areas that were not covered in the 1988 version:

- Access Control. Access can be restricted to the DIB by a set of Access Control Lists identifying each user's level of access.
- Replication. Two new protocols have been added to facilitate an effective master/shadow model for replication of data.

- Schemas. With the use of operational attributes, schema rules can be represented in the directory and can reflect the structure of the DIB and ensure a consistent structure is maintained.

In addition, a new 'Extended Information Model' with a 'User, Operational, and Administrative View' has been developed providing mechanisms to clearly describe the internal behaviour of the directory. Searching has also been improved with the ability to receive the results of large searches page-by-page. Further discussions of the 1992 extensions can be found in (Exner, 1992; Radicati, 1992b; Steedman, 1993).

3.5 Summary

Both the X.400 and X.500 standards have been developed with maximum potential for providing services to end users. The respective standards outline and define the functionalities of each service and the requirements for intra-service communications. As has been shown, the vast range of options available are promising and include a wide coverage of services.

Clearly lacking in the standards are the requirements for user interfaces to message and directory user agents. These, of course, have been purposely omitted from the standards. The OSI standards have a role in defining the application-level requirements for interoperability between the various layers. The specification of user interface guidelines for particular services has been left to the implementors of OSI products and the research community. The advantage of this is that it allows novel user interface paradigms to be applied to the standards and encourages competitive challenges. One possible disadvantage is that the end user is faced with a plethora of different and inconsistent user interfaces.

The major user interface problems arising from the outline of the X.400 and X.500 standards can be broadly catergorised as:

- X.400 message options (service elements),
- X.400 addressing, and
- X.500 directory integration for IPM UAs.

Each category provides an essential requirement for the provision of MHS services and requires significant effort in the design of suitable user interfaces. Specific problems include the following:

- The vast range of X.400 message options will overwhelm the end user if not properly presented and managed.
- The complex and arcane X.400 addressing requirements for message recipients will hinder widespread acceptance.
- The directory searching, for the retrieval of recipient X.400 addresses, requires tight integration with the UA for effective provision of messaging services.

In the future, it is highly unlikely that any new versions of the standards will attempt to address any of these major points. The standards will, however, continue to reflect the *provision* of services and not the user interfaces to those services.

Chapter 4

Related Work

4.1 Introduction

This chapter covers areas of research and development that currently exist in the design of electronic mail interfaces. A discussion of the functional needs of electronic mail systems are identified. Interfaces need to be developed to provide uninhibited access to such functions. Some early work on interfaces to a basic electronic mail system is also discussed.

An investigation on the design of X.400 UAs has identified a serious lack of specific research in the area. Some abstract guidelines are reported for UA interfaces that have mostly been developed on text-based systems. The issue of X.400 addressing is highlighted with work on concise formats for O/R addresses being developed to provide consistent means of communication. The integration of X.500 Directory Services with UAs is discussed and the lack of such services in current UAs noted.

A comprehensive review is presented of popular graphical user interfaces for electronic mail systems. A number of non-X.400 based systems are analysed with respect to their functionality with message lists, folder management, composing new messages, and addressing messages. The same issues are discussed with the limited number of X.400 based systems.

Finally, three systems are outlined which have provided the framework for the development of Iris. These include the X Window System and the OSF/Motif graphical development environment, the PP X.400 message handling system, and QUIPU X.500 directory system. A short discussion of lightweight protocol access to X.500 Directory Services is also mentioned as the main directory integration methodology for Iris.

4.2 Electronic Mail Interface Design

Designing interfaces for electronic mail systems requires an understanding of both the psychology and the utility of electronic mail. Electronic mail systems have been in use for many years and their psychological effect on the user needs to be understood for an effective inter-

face design to be achieved. Becker (1990) identifies the following characteristics of electronic mail that affect people:

- Rapid transmission with delayed feedback.
- Low context (only written words).
- Privacy while reading and composing with the results becoming public and permanent.

Interface design can be improved by understanding these and other issues and making electronic mail complement human behaviour. In some cases, the electronic mail community has established de facto standards to convey the emphasis in mail messages. A series of 'smilies' has been developed which reflect the mood of the message originator or the intended interpretation of the message. The smilies are sideways faces such as happy :-) or sad :-(inserted in the message body. The advent of multimedia mail with full text formatting will alleviate this problem.

Electronic mail systems require a minimum set of functions that enable the user to perform message communication tasks. These functions vary between disparate electronic mail systems but generally are required to be easily accessible and convenient. Electronic mail system functionality should include the following:

- read incoming messages,
- send new messages (including adequate text editing),
- reply to and forward messages,
- file messages into user-defined folders for later retrieval,
- delete and print messages, and
- notify the arrival of new messages.

Ideally, these functions should be provided with a post-office-like metaphor both visually and conceptually. Holleran & Haller (1990) provide a complete discussion of functional characteristics of communication systems. Bonsiepe (1990) discusses the design of an electronic mail system with emphasis on providing the desirable functionality and the visual aspects of the graphic design framework. Malone *et al* (1986) describes the use of 'a rich set of semi-structured message types that can form the basis of an intelligent information sharing system'. The imposing of structure within a message body will enable automatic filtering to a set of user-defined rules. However, the X.400 protocol does not specify any structure *within* each body type and hence would not be able to accommodate these ideas.

Problems with electronic mail systems in the interface, reliability, limitations, and addressing difficulties are well documented (Tarouco, 1984; Pliskin, 1989). Some of these problems can be solved with appropriate usability trials of the interface, others, such as arcane addressing, are still to be solved. Evaluation of electronic mail systems varies greatly in technique including:

- Professional usability labs (McIntosh *et al*, 1991).
- Empirical methods (time-related) of user performance (Dowell & Long, 1989).
- Analysis of errors (Akin & Rao, 1985).
- Usage monitoring (Rees & Iannella, 1990b; Rees, 1991).

Although no single method purported to be a complete evaluation solution, each identified areas of the interface that could be improved. Usability evaluations, although more expensive and time-consuming, generated a more thorough evaluation result.

Early work on elementary electronic mail interface design and evaluation (Rees & Iannella, 1990a) resulted in a functional system called COREmail with an obvious user interface. COREmail was used to teach the electronic communications section of an introductory computing course and proved useful in the evaluation of interfaces for novice users. COREmail, developed with HyperCard, and incorporating automatic usage logging, developed consistent rules for the graphical behaviour of interface objects. Recipient addressing, using scrollable lists, proved successful up to a limit of approximately 100 addresses after which a drag-n-drop technique was proposed. Figure 35 shows two screens from COREmail.

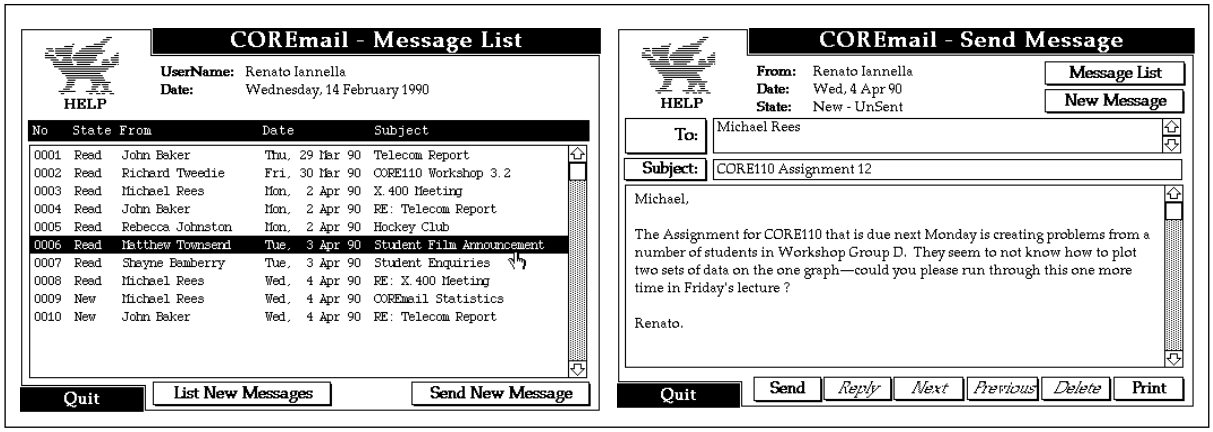


Figure 35: COREmail Message screens.

4.2.1 X.400 User Agent Design

The X.400 standard does not cover any interface design issues for UAs, only detailing the functions required by cooperating UAs. This omission has both advantages and disadvantages. The main advantage is that the interface is open to innovative design implementations and does not have to conform to standards that may be dated or dependent on past technologies. The main disadvantage is that potential users may be faced with a disparate array of inconsistent UA interfaces.

Early work on generic UA design (Thompson, 1988) derived the following requirements of the user interface:

1 User interface should not overload the user with too much information and should exhibit simplicity.
2 The degree of functionality should be tailored to user's needs and requirements in the particular circumstance.
3 Interactive mechanisms (command line, menu, or window-based) should allow the user to direct the UA for different levels of user competence, training, and personal choice.
4 A user's initial contact with he UA should be fruitful and useful in terms of successfully manipulating the desired results and success with interacting with the UA.
5 As with written communications, users expect a certain level of security and privacy for their UA messages.
6 The presentation of the UA functions should model, where possible and realistic, the real world office environment.

The above points indicate a need for common and consistent view for UA design without constraining an implementation or stifling creative design.

Other UA design issues include the archiving and retrieving of messages. Users should be able to file messages into appropriate folders for later retrieval. As with non-X.400 systems, effective management of existing messages is critical to the system usage. Plattner & Steinparz (1988) and Altomare *et al* (1989) outline some requirements for message archiving services. Prasad (1990) outlines methods to provide a unified user interface to both X.400 and Unix mail by automatically converting between the two. Unfortunately, this does reduce the functionality of X.400 mail down to the lowest common denominator.

Previous UA design has concentrated on text-based interfaces for IPM services. Chapman (1986) outlines problems faced in UA design due to the high functionality of X.400 and the potentially long data fields that are difficult to handle on small text-based screens. Other text-based problems include the requirement of the user to specify options in either full-text or requiring extraneous keyboard traversal of the display, both burdening the users load. Kairi (1987) and Kairi & Barnard (1988) discuss a UA that provides full X.400 conformance but limited by a text-based UA. The adoption of a graphical interface for UAs alleviates some of the problems with the large range of X.400 options. Figure 36 shows an example of a text-based UA screen in which the user is required to enter full message options.

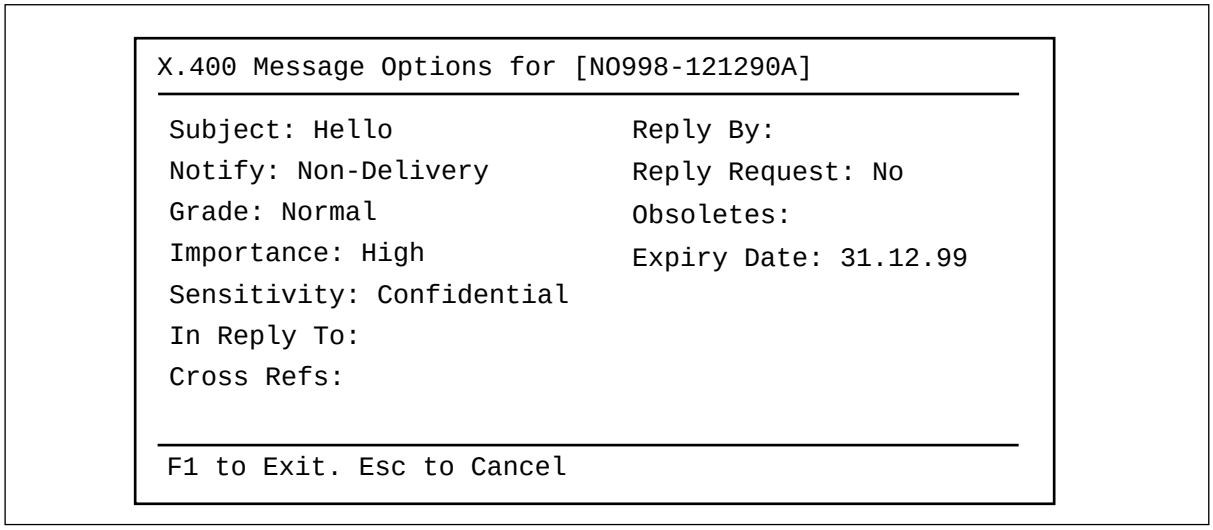


Figure 36: UA Text-Based Screen Example

4.2.2 X.400 Naming and Addressing

Naming and addressing in X.400 has been the major contentious issue for the standard. Electronic mail addresses, and the lack of standardisation, was one of the reasons for the introduction of the X.400 standard. The structure used for O/R addresses reflects a bias towards the routing of the messages.

Almost all existing UAs require the O/R address to be entered in full text including the attribute qualifiers. For example:

```
To: /G=Fred/S=Smith/O=Bond/PRMD=OZ/ADMD=Telememo/C=AU
```

Jeffree (1989) describes the general principles behind the X.400 addressing structure and concludes the need for directory support for names and routing information.

Naming and addressing is not a new problem for the communications domain and has always been a debatable issue. Norman (1992) outlines the reasons why there will never be a single, simple solution to the complex naming problem including the legal, moral, religious, and cul-

tural factors that vary radically across the world. The issue of privacy is also of concern with different views as to what information about a person should be made public. Even though national standard bodies have adopted the X.400 standards for naming and addressing (Standards Australia, 1991) they still lack any real solutions to the user's problem of comprehending the O/R address structures.

An area receiving a large amount of attention is in a shorthand notation for X.400 addresses (Grimm & Heagerty, 1989). The aim is to present a single and precise format for the consistent exchange of X.400 addresses. The goal of the notation used is to be brief, simple, and unambiguous to understand. Bevan (1991a) has performed human factors studies on three proposed formats for the representation of X.400 addresses with the objective to find a consistent format to use on business cards. The three formats are shown in Table 16.

Table 16: Proposed X.400 Address Formats

Format Name	Format Style
Self Explanatory	Country (C): GB ADMD: Gold 400 PRMD: BBank Organisation (O): Bank of Britain Org. Unit (OU1): DP Dept Surname (S): Smith Given name (G): John
Labelled	C=gb; A=gold 400; P=bbank; O= bank of britain; OU1=dp dept; S=smith; G=john
Concise	john/smith/dp dept/bank of britain/bbank/gold 400/gb

The studies showed that the Labelled format was a good compromise between the error prone Concise format and the slower Self Explanatory format. Two issues still to be resolved are the delimiters between fields (semicolon or forward-slash) and in which sequence the attribute elements should be listed. However, CCITT has adopted this format as an appendix to the X.400 standard (CCITT, 1993).

4.2.3 X.400 and X.500 Integration

One solution to the X.400 naming and addressing problem is integrating X.500 Directory Services into UAs. Little research has been reported in this area as current systems rely on the use of separate applications for X.500 queries and the user is responsible for the transfer of addresses into the UA. This situation still relies on the user having knowledge of X.400 O/R address structures. X.500 integration can also be used to hide the underlying complexity of MHS and to make control of data more reasonable and functional (Kille, 1990).

Using an X.500 directory, a complicated X.400 O/R address such as;

C=AU; ADMD=Telememo; PRMD=OZ; O=Bond; S=Smith; G=Fred;

could be replaced with a directory entry as;

Fred Smith, Bond University, AU

Apart from the two character country code, which most DUAs should be able to convert into full country names, the directory entry is easy to remember and reasonably obvious to novice users.

The key to X.500 integration is to allow the UA to perform the search, given a directory entry, and lookup the corresponding O/R address. If no such entry exists or the entry has no O/R address attribute, then appropriate errors are indicated. The directory search technique would use appropriate matching strategies (Wickham, 1991) to return the best results, thus allowing users to abbreviate personal and organisational names.

Hinrichs & Prinz (1991) discuss an example of application-specific integration of X.500 into a *Media Assistant*. The Media Assistant allows the user to search for an entry, then communicate with that person via four access methods: electronic mail, fax, paper mail, and telephone. The Media Assistant then prepares the communication method and reads the appropriate attributes from the directory and prompts the user for the remainder of the message.

Directory services integration into electronic mail systems is not dependent on X.500 or X.400. Persky (1992) discusses a non-X.500 DSA that integrates into a non-X.400 MHS providing address lookups. Future plans for such systems is to move towards standards compliance.

4.3 Review of Existing Electronic Mail Interfaces

Existing graphical user interfaces to electronic mail systems provide a rich source of ideas and inspiration for new interfaces. There are common aspects of all electronic mail systems that should be followed to provide consistency across the application domain. Some systems also provide novel ways to present information to the user and provide an aesthetically pleasing interface. Graphical electronic mail systems have gained greater acceptance over text-based systems by providing all the advantages of direct manipulation interfaces with the increased functionality of electronic communication. Farris (1991) reviews a number of graphical electronic mail products and the advantages they provide over text-based Unix mail applications.

The main three areas of interest to this review are:

1 message lists and folders,
2 message composition, and
3 message addressing

across both X.400 and non-X.400 systems

4.3.1 Non-X.400 System Interfaces

A review of non-X.400 systems highlights advances in graphical interfaces that have proved successful in many organisations. Mainly concentrating at the personal desktop computer, these electronic mail systems have provided intuitive graphical interfaces and effective increases to electronic communications.

Under review are six common electronic mail systems in use across a wide variety of computer systems. These include:

1 Microsoft Mail (Microsoft Corporation)
2 QuickMail (CE Software Inc.)
3 cc:Mail (Lotus Development Corporation)
4 Z-Mail (Z-Code Software Corporation)
5 NeXTmail (NeXT Computer Inc.)
6 Xmh (MIT X Consortium)

Message Lists & Folders

Message lists and folders provide summary information of the current list of messages and the message folders that the users has previously created. The message list consists of one line of text per message usually indicating the originator, subject and date of the message. Other feedback may be used to indicate new messages or other pertinent information. Message folders are usually created by the user and can be used to file messages for later retrieval. There is usually at least one system-dependent folder (sometimes called *inbox* or *mailbox*) that is the default folder that holds new and current messages (to be filed).

Figure 37 shows example message list screens from Microsoft Mail and QuickMail. To indicate new messages, Microsoft Mail highlights the originator (From) in bold and QuickMail shows a grey message icon. Message icons are used to indicate different types of messages (eg receipt-notifications) and messages with documents enclosed. Both systems list the message folders on the bottom half of the window and use direct-manipulation to open folders and to file messages. The latter can also be achieved by selecting the message folder from a dialog box presenting a list of folders. When reading a message, a new window is displayed showing the message heading fields and body.

The benefit of these two interfaces is that the user has direct control over the messages and folders. The user can also visually interpret the state of the messages and review the summary information about each.

A slightly different setup is shown in Figure 38 in which cc:Mail uses two separate windows to show the list of message folders and the messages contained in those folders. Similar techniques are used to indicate new messages, different types of messages, filing messages, and reading messages in separate windows.

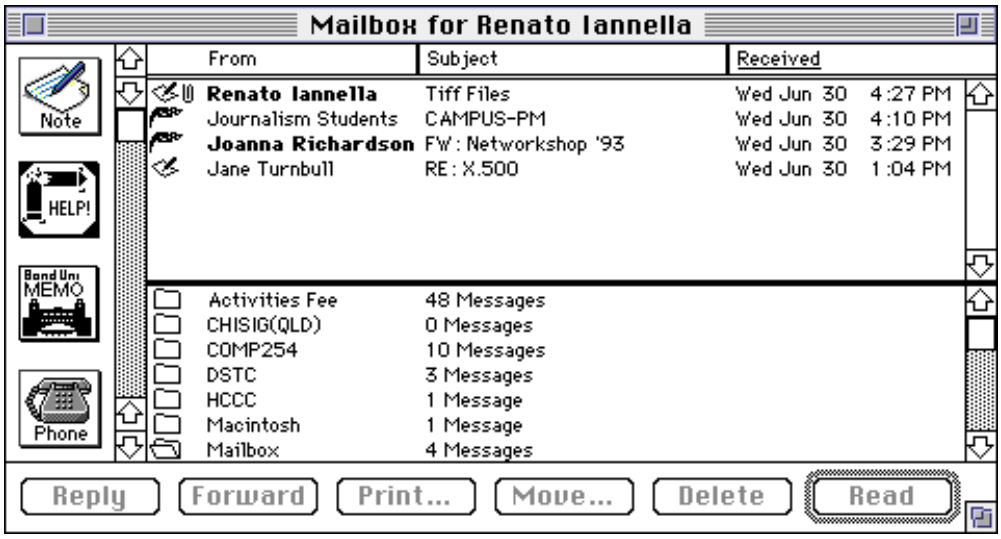
The main disadvantage with the cc:Mail approach is that the user is now forced to navigate and manipulate two separate windows. This will lead to some initial usability difficulties.

NeXTmail (see Figure 39) uses a similar technique with separate message folders window (called Mailboxes) but displays the message in the same window as the message list. When a message is selected from the message list, the message details are displayed on the bottom half of the window.

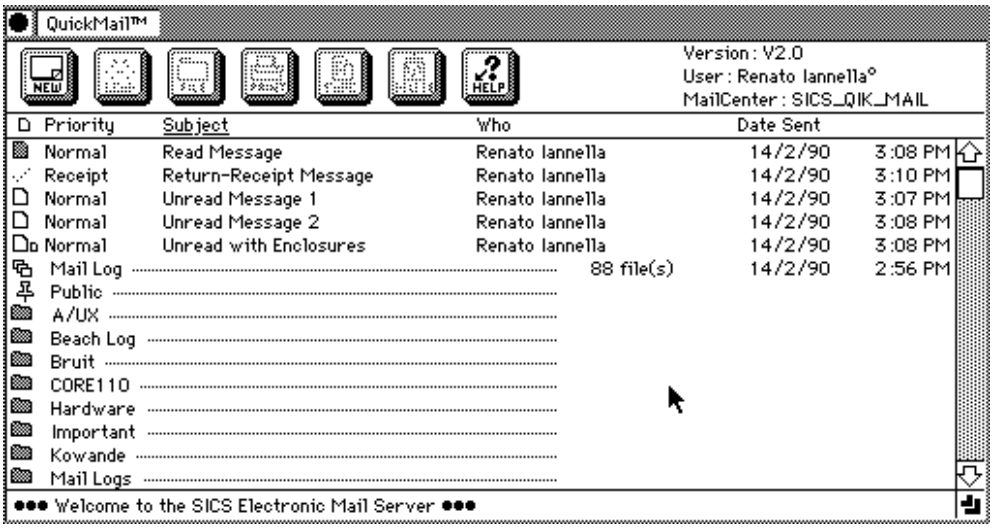
One major disadvantage of the NeXTmail system is that the only one message can be viewed at a time. Also, similar to cc:Mail, two windows need to be manipulated by the user.

Xmh was one of the first graphical interfaces for messaging systems under the X Window System. As a result, it has a very simple and primitive window interface with four panes as shown in Figure 40. The top pane is a series of pull-down menus and the second pane is the list of message folders. The third pane lists the one-line summary of the messages (of the currently selected folder). The bottom pane displays the heading fields and text body of the currently selected message.

The Xmh approach—a single window for all the information—does have some disadvantages as noted with NeXTmail. However, the simplicity is an added bonus, especially for first time Unix mail users.



Microsoft Mail



QuickMail

Figure 37: Message List and Folders

More sophisticated interfaces has since emerged under the X Window System. Figure 41 shows the message list screen from Z-Mail. Message folders, in this case, have to be explicitly specified or chosen from a separate dialog list.

The Z-Mail approach, although commendable for a Unix mailer, does little to hide the underlying legacy system. New users would find it difficult to interpret the directory structure and layout of the screen.

Message Composition

Composing new messages in all of the systems involves selecting the new message option and results in a new window being created. The new message option may also involve selecting a different type of message form. For example, the list of message icons on the left of the Microsoft Mail window (see Figure 37) shows different message templates. However, for the Iris prototype, single text-only body parts are assumed.

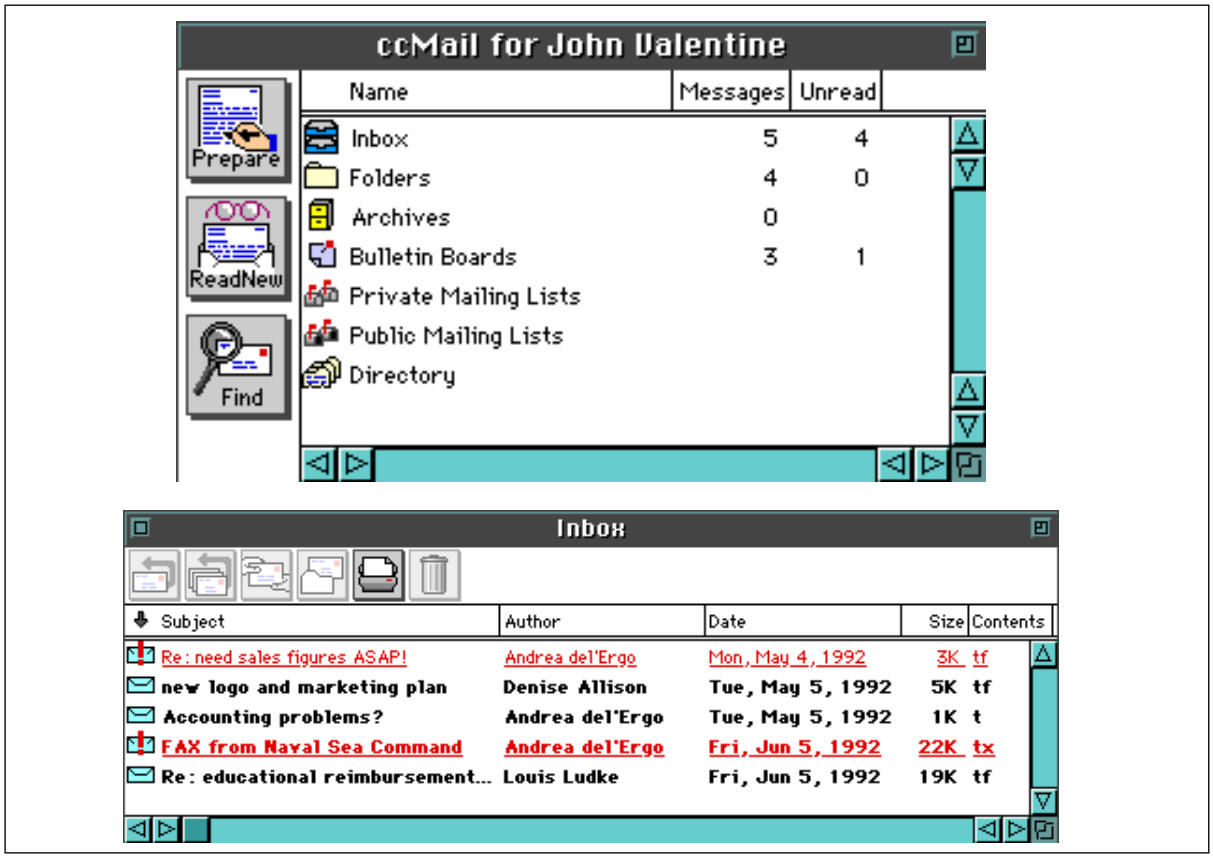


Figure 38: cc:Mail Message Folders and List

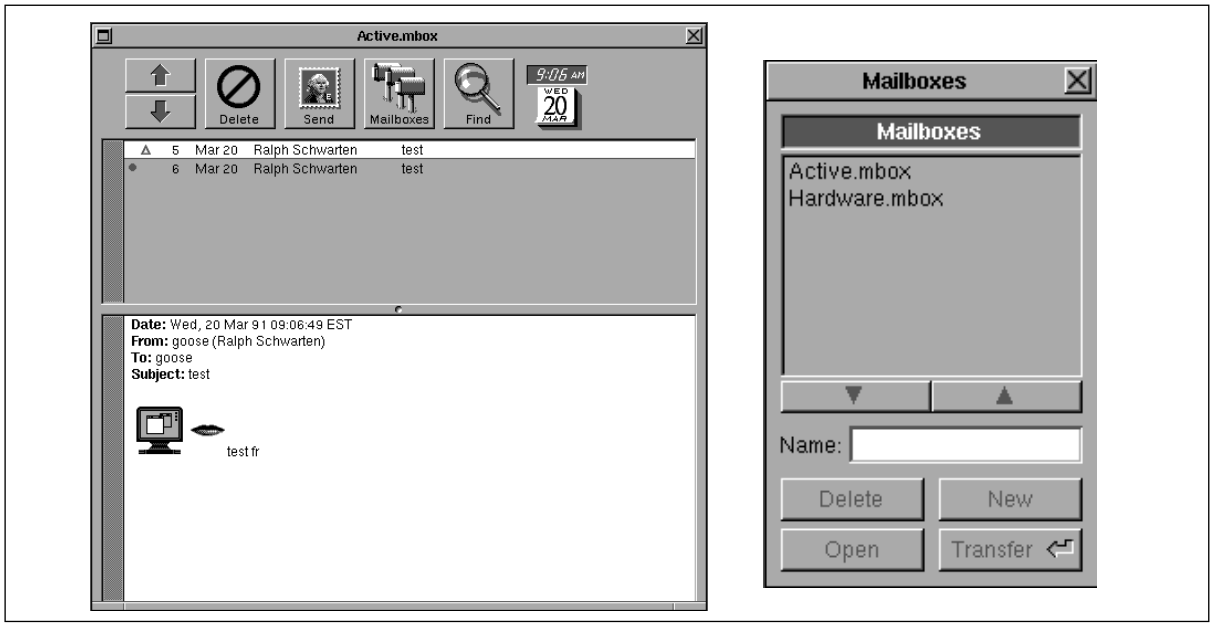


Figure 39: NeXTmail Message List and Folders

Once the new composition window is created, the user may then set message options (eg receipt notifications) and then enter the body of the message. In some cases, external documents may also be attached to the message, or other media body parts entered (eg voice annotations in NeXTmail). Overall, this process is similar and consistent across most electronic mail applications. Figure 42 shows a sample of the message composition windows from each system reviewed.

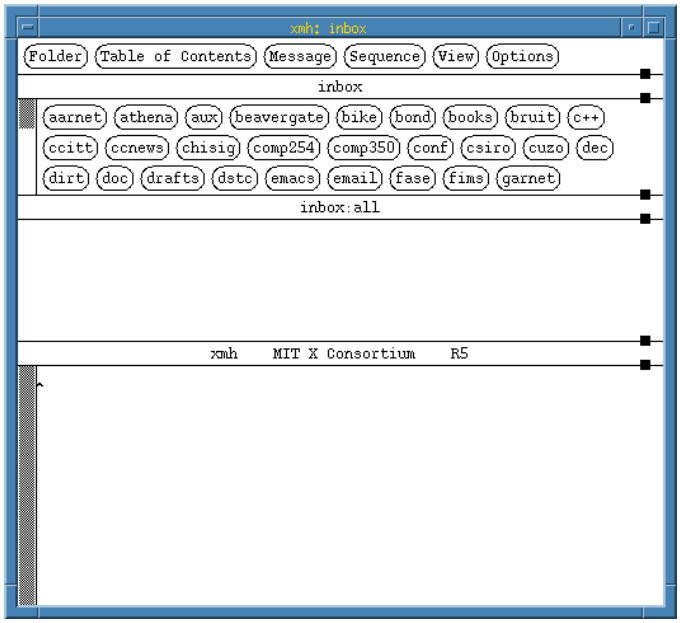


Figure 40: Xmh Message List and Folders

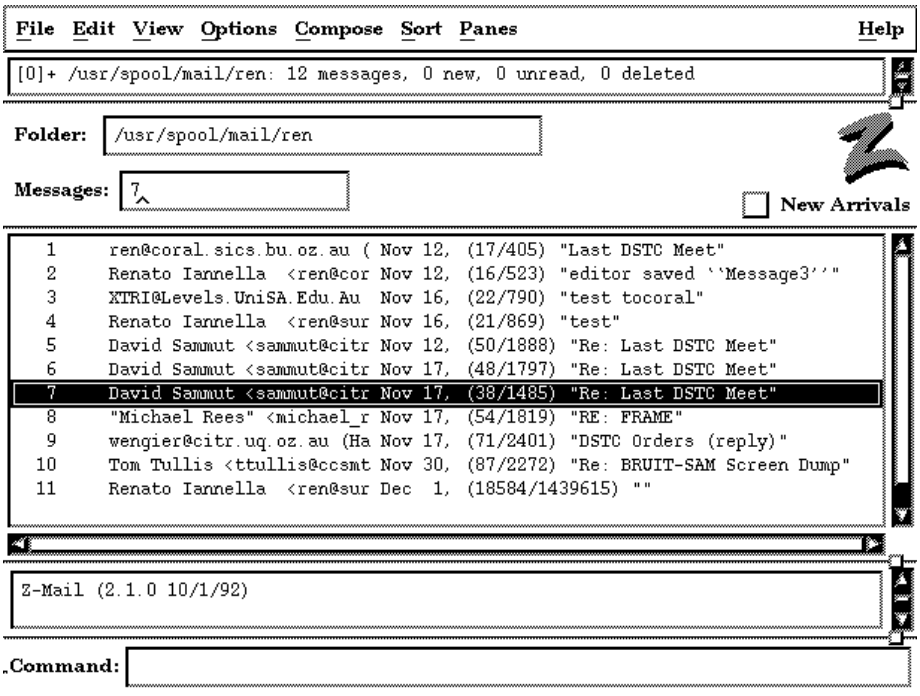


Figure 41: Z-Mail Message List Screen

The four composition screens from Microsoft, QuickMail, cc:Mail, and NeXTmail utilise an iconic interface, with textual subscripts, for the selection of message options. Microsoft Mail message options are entered via an extra window that is displayed when the Options button is selected. This method saves on screen real estate but does introduce another level of interaction. The layout of the screens are consistent, although the cc:Mail screen is somewhat cluttered and not obvious (eg the priority level). Xmh offers little advantage over text-based mail composition, apart from simple editing with the cursor, which could lead to serious problems if the 'To:' line is deleted. Z-Mail offers a more appropriate composition window for Unix us-

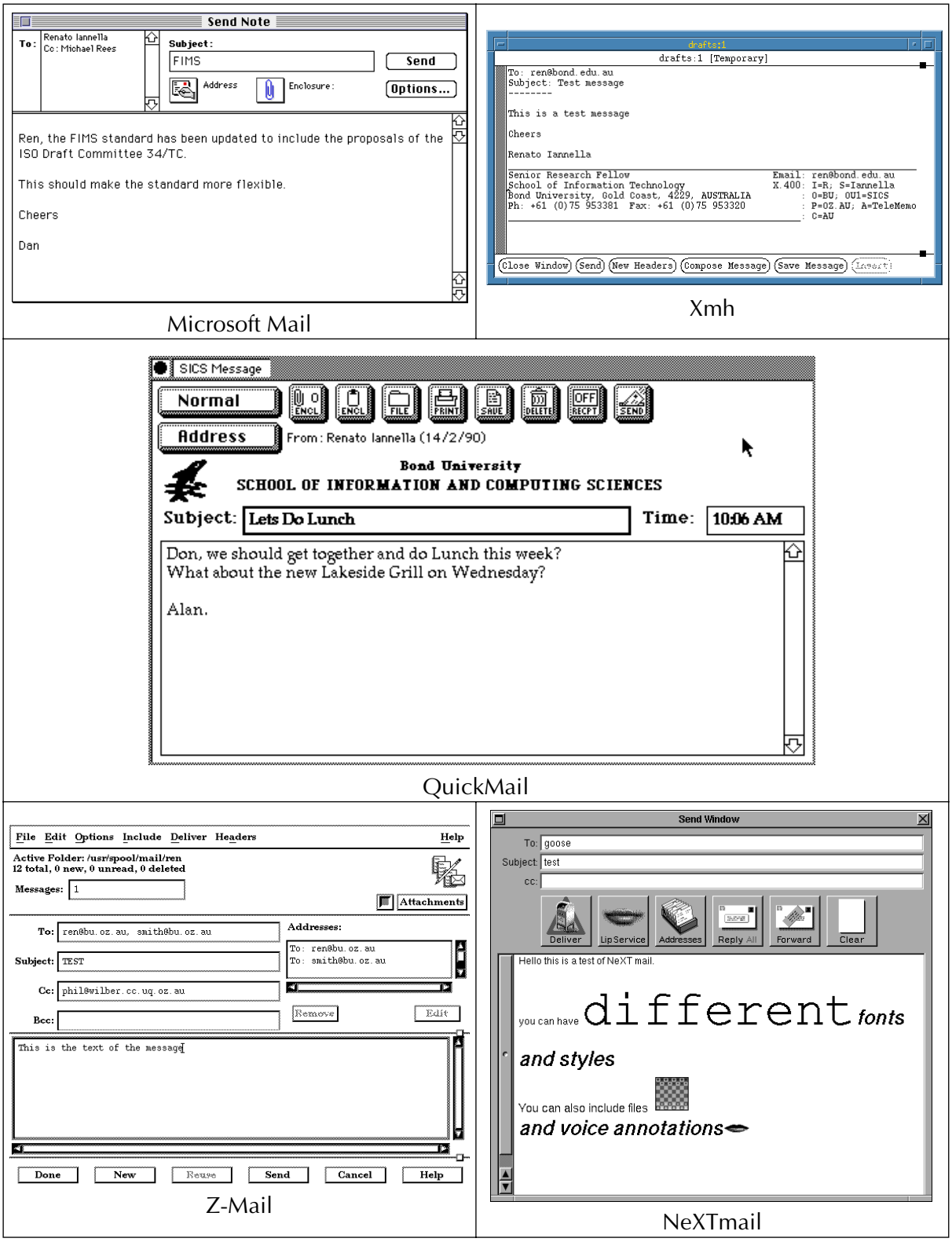
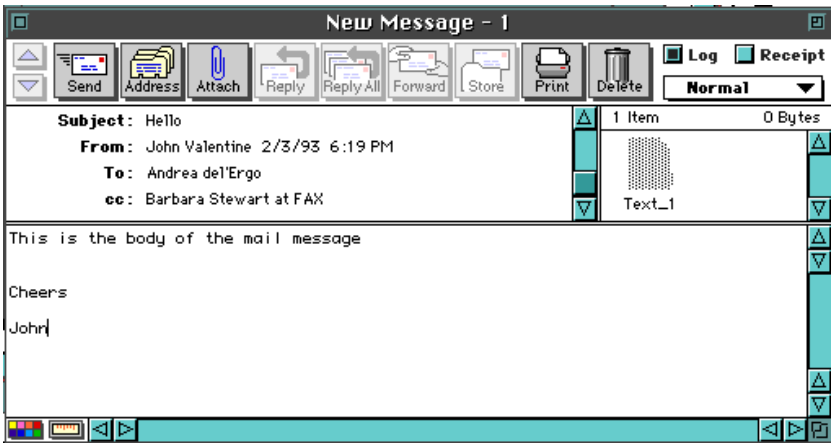


Figure 42: Message Composition Windows



cc:Mail

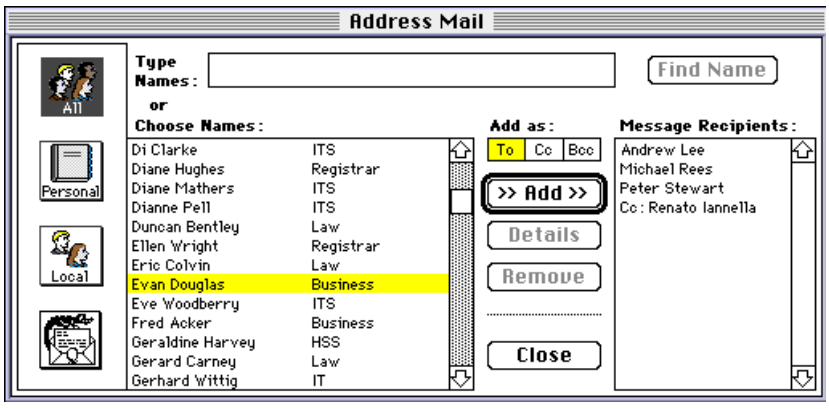
Figure 42: Message Composition Windows *(continued)*

ers, although the single line text fields for recipient specification are difficult to manipulate. Also, the positioning of some of the buttons (eg Remove and Edit) are unclear.

Message Addressing

Ease of message addressing is the decisive element to a successful electronic mail system. All the systems in this review, except Xmh which requires textual entry of recipients, provide lists of recipients to use in addressing. Systems running a local mail server (such as Microsoft Mail, QuickMail, and cc:Mail) have access to the database of local recipients as well as means of setting up personal 'address books' for recipients outside the local mail server (or gatewayed to other systems). Other systems (such as Z-Mail and NeXTmail) allow the user to setup alias entries to potential recipients. Also available is the option to setup groups of recipients (distribution lists) into single alias names.

All of the addressing schemes allow the user to select a recipient category as either primary, carbon copy, or blind carbon copy. (NeXTmail does not support the blind carbon copy recipients category.) Figure 43 shows a sample of the message addressing windows from each system.



Microsoft Mail

Figure 43: Message Addressing Windows

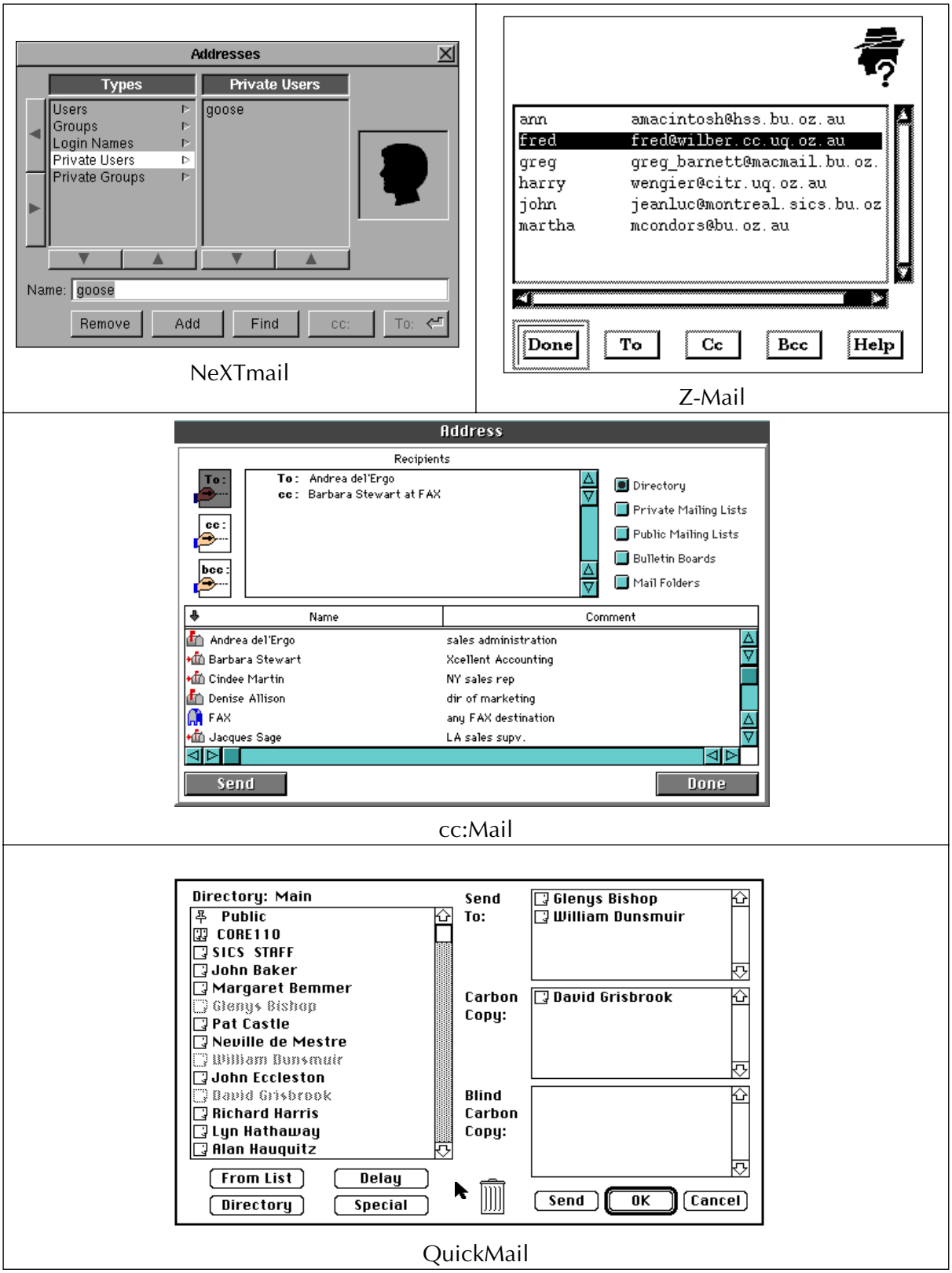


Figure 43: Message Addressing Windows *(continued)*

Microsoft Mail has a series of icons on the left of the window indicating different recipient lists and for external addresses. The recipient list can also be narrowed down by specifying parts of the recipient name. Mail groups (distribution lists) are displayed in bold. Double-clicking on the recipient name (or selecting the '>>Add>>' button) moves the recipient name across to

the message recipients list. A similar technique is used in cc:Mail which also includes various icons to represent different types of message recipients (eg fax numbers).

QuickMail utilises the drag-n-drop metaphor and allows the user to drag recipients across to the appropriate recipient category box. Mail groups are displayed with a double-faced icon. Z-Mail uses a simple list of (predefined) recipients and allows the user to choose the category of the recipient via a series of buttons.

Overall, each of the direct manipulation addressing windows are effective and provide the user with simple editing of addresses. Microsoft, cc:Mail, and NeXTmail also allow the user to create private address groups, which enables the user to categorise the recipients. The other systems can be cumbersome to use when having to scroll through a large list. However, Microsoft Mail's use of specifying the first few letters of the address, to narrow the list, is a good alternative. The different icons used for personal entries in cc:Mail, and a number of unclear toggle buttons (eg Mail Folders on the Addressing window) would lead to some usability problems.

4.3.2 X.400 System Interfaces

There has been little development of graphical user interfaces for X.400 UAs and only a small number of commercial UAs are available. RetixMail[1] and the recently released XT-MUA[2] are two examples of commercial X.400 UAs utilising graphical user interfaces. RetixMail is available on Macintosh and Microsoft Windows environments and presents a consistent interface to X.400 services. RetixMail only supports the 1984 version of X.400. RetixMail is dependent on an MTA running on a dedicated machine with clients connected with Network File System. XT-MUA is available for X Windows System machines running Unix and supports both 1984 and 1988 version of X.400. XT-MUA offers a flexible interface to X.400 services including cut-and-paste support with an X.500 DUA.

XMUA[3] is a publicly available X.400 UA running under the X Window environment. The graphical interface is very primitive and unsophisticated. Figure 44 shows an example of the main window from XMUA in which all status information, message lists, and results are displayed in a large scrolling list.

In composing messages, XMUA uses a series of dialogs in which the user is requested to select options for each possible service element. This can sometimes be overwhelming and unnecessary and utilises little of the available graphical user interface components. For example, Figure 45 shows a series of dialogs in which the user is asked to select various options. In this case, the user has selected the Primary recipients composition field, the IA5 text Body part, RN and NRN Report requests, Non-urgent Priority value, and a False Reply request. A further series of dialogs appear in order to complete the IPM message submission task.

Addressing of messages in XMUA involves the selection of a predefined recipient name or the creation of a new address. In the latter case, the user enters a shortform name of the recipient and then enters the O/R address into an address form dialog (see Figure 46 on page 100). Overall, XMUA is an example of an 'interface-independent' application. That is, little use is

1. RetixMail is a trademark of Retix.
2. XT-MUA is a trademark of NeXor Ltd.
3. XMUA was developed at INESC-Centro de Comunicacoes em Ambientes Empresariais, Portugal.

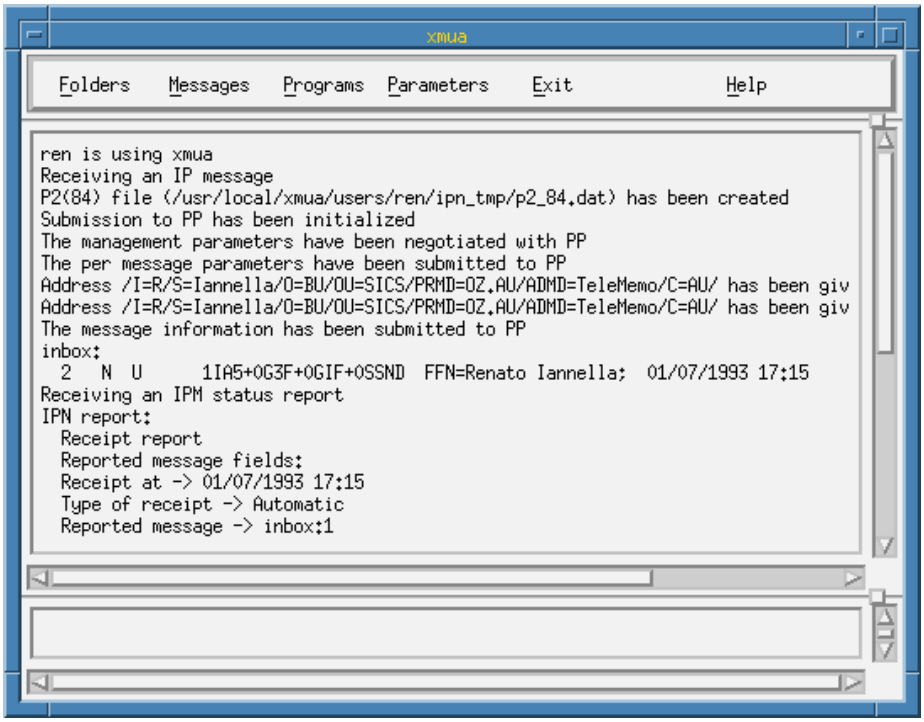


Figure 44: XMUA Main Window

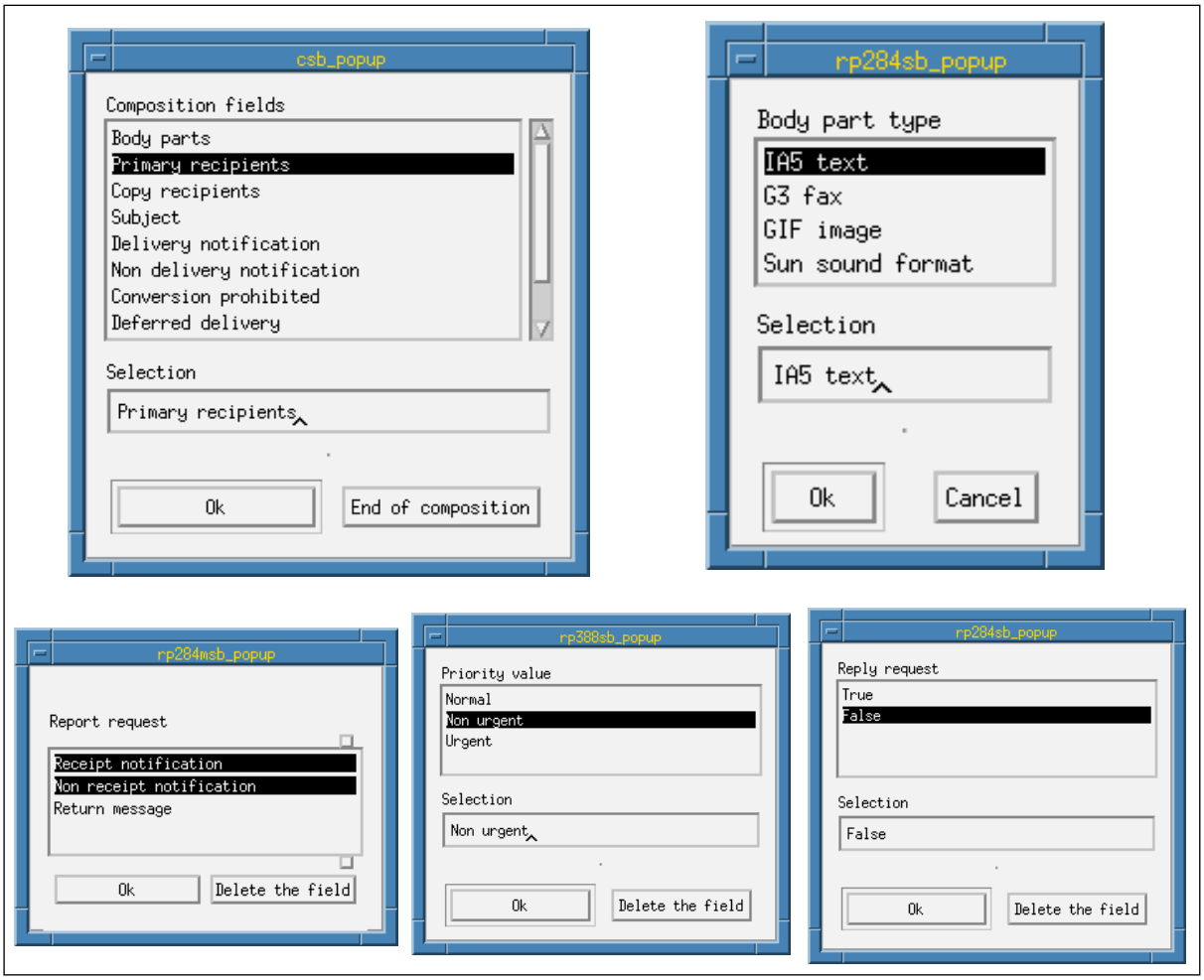


Figure 45: XMUA Message Composition Dialogs

made of the rich interface components available to enhance the user's capability with X.400 messaging systems.

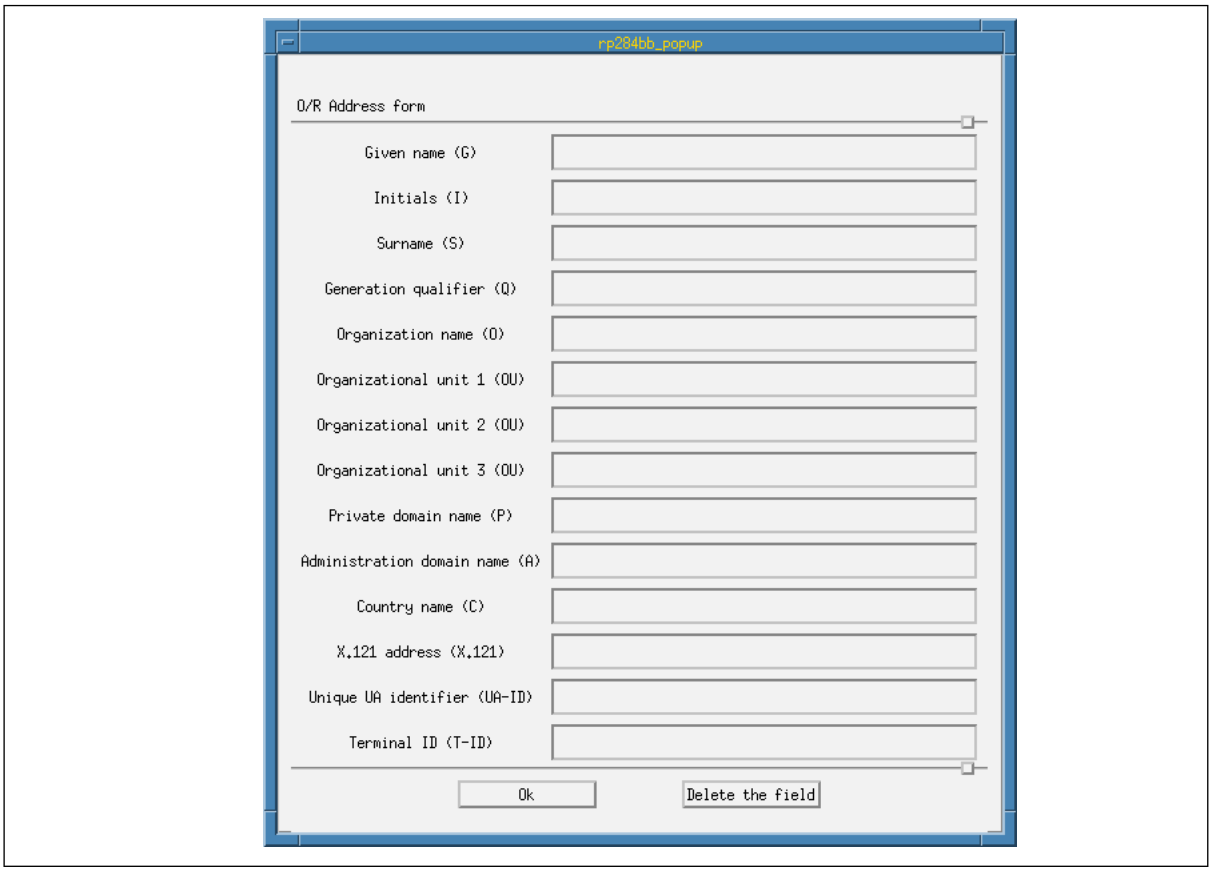


Figure 46: XMUA O/R Address Form Dialog

4.4 X Window System Overview

The X Window System was developed at the Massachusetts Institute of Technology as part of the campus-wide Project Athena[4] (Scheifler & Gettys, 1987). The X Window System, commonly referred to as 'X', provides a network-oriented windowing system designed for graphical display workstations. The main areas that X addresses are the use of hardware-level techniques for maximum performance and to generate a common programming interface across heterogeneous platforms for portability.

X uses a client/server paradigm and is designed to be device independent by utilising a network protocol. The X *server* is responsible for performing the graphic operations on a workstation and the X *client* is the application program that connects to the server using the X protocol. X's client/server nomenclature is reversed with the server usually running on the users desktop workstation and the client on a distant 'server' machine. Both the client and server may be running on the same machine or across a network on dissimilar hardware. With device independence, the client could be displayed on any hardware with a capable X server and network connections.

The actions of the user (via mouse, keyboard, or other input devices) interacting with the server are translated into the X protocol and transmitted to the machine running the client, which

4. In conjunction with Digital Equipment Corporation and International Business Machines

interprets and processes the actions. This may include sending requests to the server to update the current display on the workstation. X is heavily dependent on the event-style of programming and generates large amounts of X protocol requests to facilitate the graphical windowing interface.

On top of the X protocol, is an extra layer called Xlib, which provides a higher level interface to the windowing facilities of X. At this level, X applications can be developed with Xlib also providing abundant graphics drawing routines. On top of the Xlib layer is the X Toolkit which provides a simplified access to the Xlib routines. The X Toolkit layer also provides two logical components called the *intrinsics* and *widgets*. A widget is an abstraction of a graphical object (such as a button or scrollbar) that can be created, manipulated, and mapped to the users display. Appropriate libraries of widgets have been developed (eg Athena widget set) providing a rich set of user interface objects and allowing rapid application development. The intrinsics provide the support for the creation of user-defined widget sets. The intrinsics are an object-based framework enabling a class hierarchy of widgets to inherit associated sets of resources.

The X Toolkit layer provides substantial code savings over Xlib. Rosenthal (1988) reports that the ubiquitous 'hello world' program takes 40 lines of Xlib code versus 5 lines of Toolkit code. X also has been reported to be superior for run-time performance, compilation time, configurability, and program development to other graphical environments (Neelamkavil & Teo, 1991). Aiding in the efficiency of X is the *gadget* which is identical to its corresponding widget but uses less resources. The X layers have C language bindings but others languages are available. Figure 47 shows a summary of the X Window System architecture.

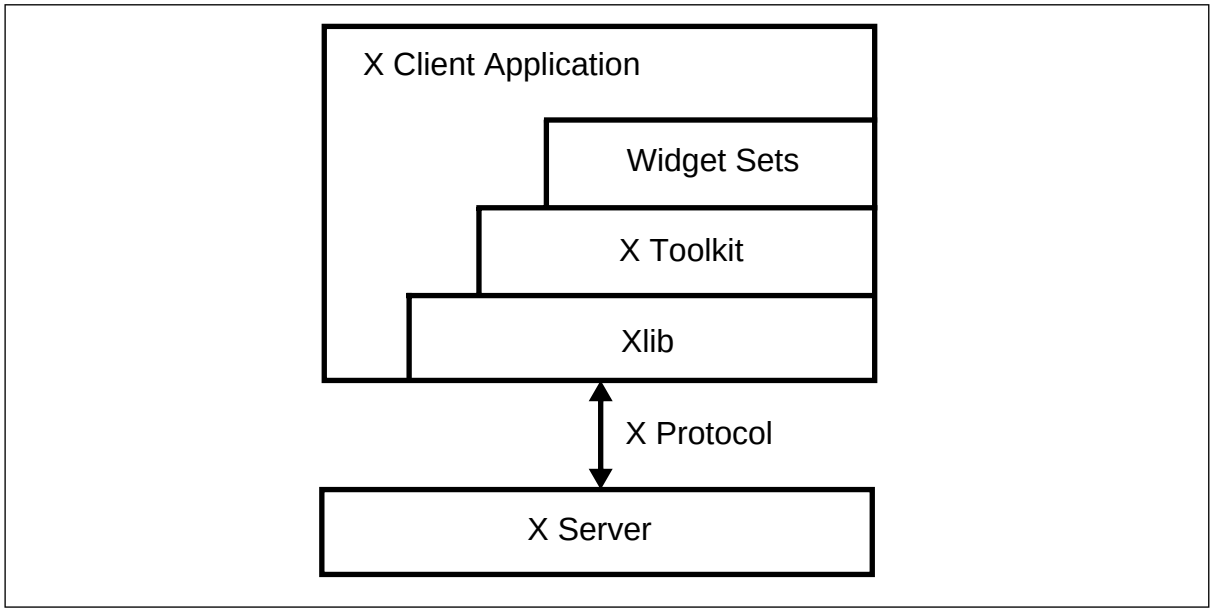


Figure 47: X Window System Architecture

One of the most successful widget sets developed for X is the Open Software Foundation's Motif (OSF/Motif). OSF/Motif is rapidly becoming the de facto standard for the X environment[5]. The OSF/Motif widget set was deliberately developed with a unique three-dimensional look to add to the visual appeal, consistency, and direct manipulation metaphor (Kobara, 1991) OSF/Motif was also developed to adhere to IBM's Common User Access behavioural

standard (see 'Common User Access Guidelines' on page 40) and includes its own style guide, user guide, and programmers guide (OSF, 1990a).

OSF/Motif provides a rich set of user interface widgets including:

- Labels
- Pushbuttons, Checkboxes, Radiobuttons, Drawnbuttons
- Arrows, Separators, Frames, Scales, Scrollbars
- Paned Windows, Scrolled Windows,
- RowColumns, Forms, Bulletin Boards
- Lists, Scrolled Lists
- Text, Scrolled Text
- Menubars, Pulldown Menus, Option Menus, Cascading Menus, Popup Menus
- Dialog boxes—Working, Warning, Question, Message, Error, Information, File Selection

OSF/Motif also provides convenience routines to enhance the customisability of colours and fonts and a Window Manager to augment the user's X environment. Figure 48 shows a sample of the OSF/Motif widget set.

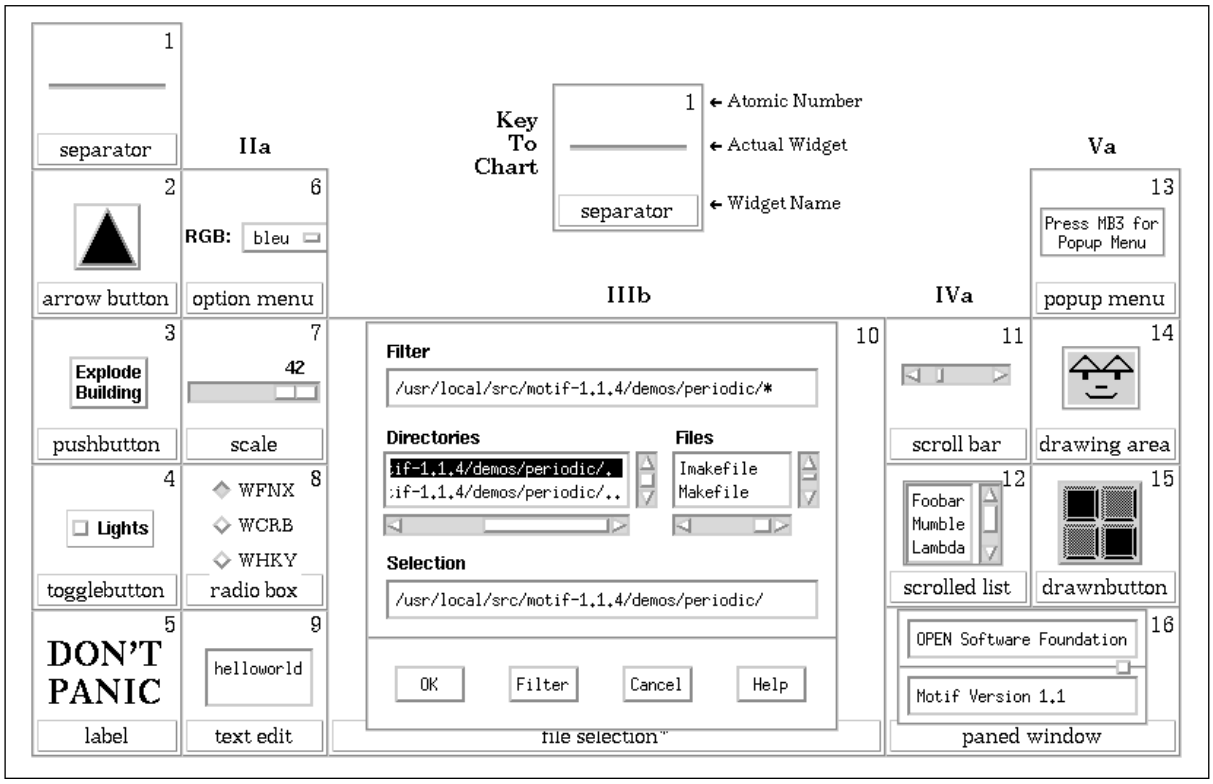


Figure 48: OSF/Motif Widget Set

OSF/Motif programs can be developed using the C language routines of the widget set or the User Interface Language (UIL). UIL is a textual descriptive language of the interface widgets that is compiled separately from the main application. At run-time, the UIL compiled file is

5. The Common Open Systems Environment (COSE) announcement on June 8th 1993 adopted OSF/Motif as the standard interface. COSE members include SUN Microsystems, whose OpenLook interface was seen as the major competitor to OSF/Motif, Hewlett Packard, IBM, Santa Cruz Operation, and Univel and is endorsed by other large Unix vendors such as Digital Equipment Corporation.

read and the widgets created along with the main application. The main benefit of using UIL is in separating the application code from the interface and machine independence.

4.5 PP X.400 MHS Overview

PP is an openly available implementation of the X.400 Message Handling System (Kille & Onions, 1991). PP is part of the ISODE (ISO Development Environment) package (Rose, 1990). PP is a message switch MTA that supports a wide range of protocols including conversion between multiple protocols. PP provides a high level of connectivity and gatewaying facilities with a extensible architecture critical to the success of a global backbone messaging service.

PP substantially supports the X.400 Message Transfer Services for both the 1984 and 1988 versions. The transport of these services is supported over OSI stacks including X.25, TCP/IP, and connectionless network services. The key supported X.400 services include:

- Deferred Delivery
- Receipt and Non-Receipt Notifications
- Message Redirection
- Distributions Lists
- Message Priority
- Probes

PP also includes a complete implementation of SMTP and other RFC 822 protocols as these are widely deployed in the research community. The issue of mapping between X.400 and RFC 822 is also addressed. The addition of new protocols is supported by PP's protocol independent framework with the ability to add extra protocol 'channels'. PP also supports a nonstandard access to facsimile. In addition to protocol conversion, PP also allows content conversion of message body parts including different character sets.

PP provides extensive configuration options including a mechanism to provide message authorisation at different levels. Authorisation may reflect the organisational policy and complete accountability is possible with comprehensive logging of services. PP is designed to support high levels of message traffic with a reported limit of approximately 100,000 messages per day (Hardcastle-Kille, 1992d) which increases with faster hardware.

A major feature of PP is the management facilities, which are critical to message service deployment. PP has dynamic control over the scheduling and control of messages with a single queue manager optimising the MTA connections. Figure 49 shows an illustration of the PP 'MTA Console' monitoring and management tool.

PP is deployed in several hundred sites around the world with an increasing trend to use PP as the 'mail hub' MTA providing uniform access to a heterogeneous collection of mail systems. PP does not include any User Agents but many existing RFC 822 based interfaces are supported with a plans to support MIME interfaces. PP APIs have been published to encourage UAs to be developed.

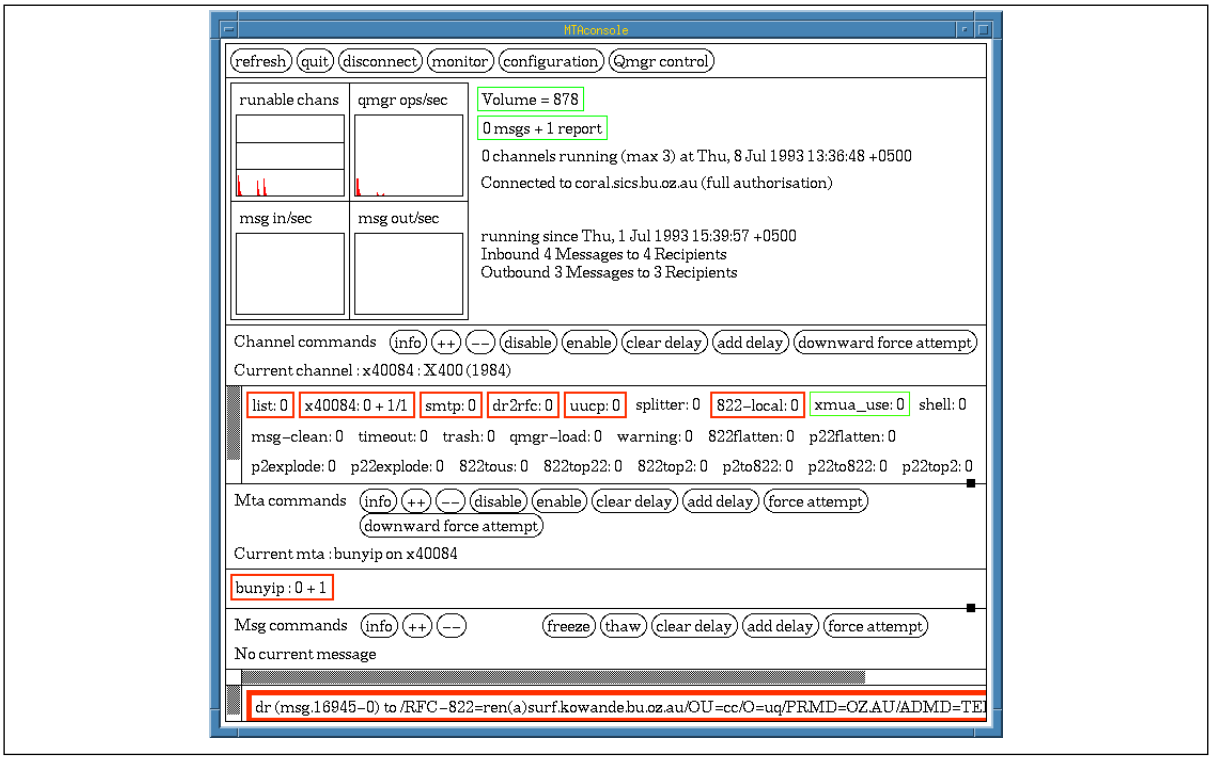


Figure 49: PP MTA Console Screen

Future plans for PP include:

- greater support for X.500 Directory Services,
- integration with network management services,
- simplified configuration, and
- greater conformance testing.

A complete discussion of PP can be found in (Kille, 1991; Hardcastle-Kille, 1992d).

4.6 QUIPU X.500 Directory Services Overview

QUIPU[6] is an openly available implementation of the X.500 Directory Services (Robbins & Kille, 1991). QUIPU is part of the ISODE (ISO Development Environment) package (Rose, 1990). QUIPU use a memory-oriented database in simple text format which is read from disk at start-up. The basic database structures are supplemented with indexes that improve the performance (reading and searching entries) of the DSA.

6. QUIPU (pronounced *kwip-ooo*) was a large rope used by the Incas of Peru to store information on a set of carefully knotted strings with coloured thread in a specific manner.

QUIPU fully implements the X.500 DAP and DSP protocols and interworks with other OSI directory implementations. QUIPU also includes a number of extensions:

- A new schema to handle the wider range of additional attributes and object class in the Internet.
- Operation over TCP/IP networks.
- Replication of information to operate more efficiently in particular for the 'top level' DIT.
- Access control to read and modify operations and mechanisms to control listing and searching of the entries.

Cooperating QUIPU DSAs use a modified DSP protocol with the above extensions. The QUIPU DSAs are themselves registered in the DIT as an Application Context entry. QUIPU also provides tools for the management of the distributed directory and APIs for the development of DUAs.

QUIPU is not intended for large scale systems (ie millions of entries per DSA) but has a very successful deployment activity (see Table 17).

Table 17: QUIPU Deployment Statistics (as of April 1993)

Countries	30
DSAs	482
Organisations	2,760
Entries	1,078,003

The QUIPU database is a series of structured text files and uses the Unix directory structure to mimic the DIT. Each text file, called the Entry Data Block (EDB), consists of entries for each object at that level in the DIT. Each entry is a attribute/value pair and is separated with a blank line. Table 18 shows an example for one entry in an EDB file.

Table 18: QUIPU EDB Example Entry

```
cn=Bart Simpson
objectClass= top & person & organizationalPerson & quipuObject
surname=Simpson
mail=Bart_Simpson@springfield.usa
description=Tutor
TelephoneNumber= +61 75 95 9999
title=Mr
```

The QUIPU directory structure has a direct mapping to the DIT that the DSA holds responsibility for. Figure 50 shows a graphical representation of the Bond University DIT structure. Under the organisation level, *o=BU*, the EDB file contains information of the organisational units. These are shown as *ou=Law*, *ou=IT*, etc directories. Under these directories, each EDB file contains information on the staff and any subordinate organisational units.

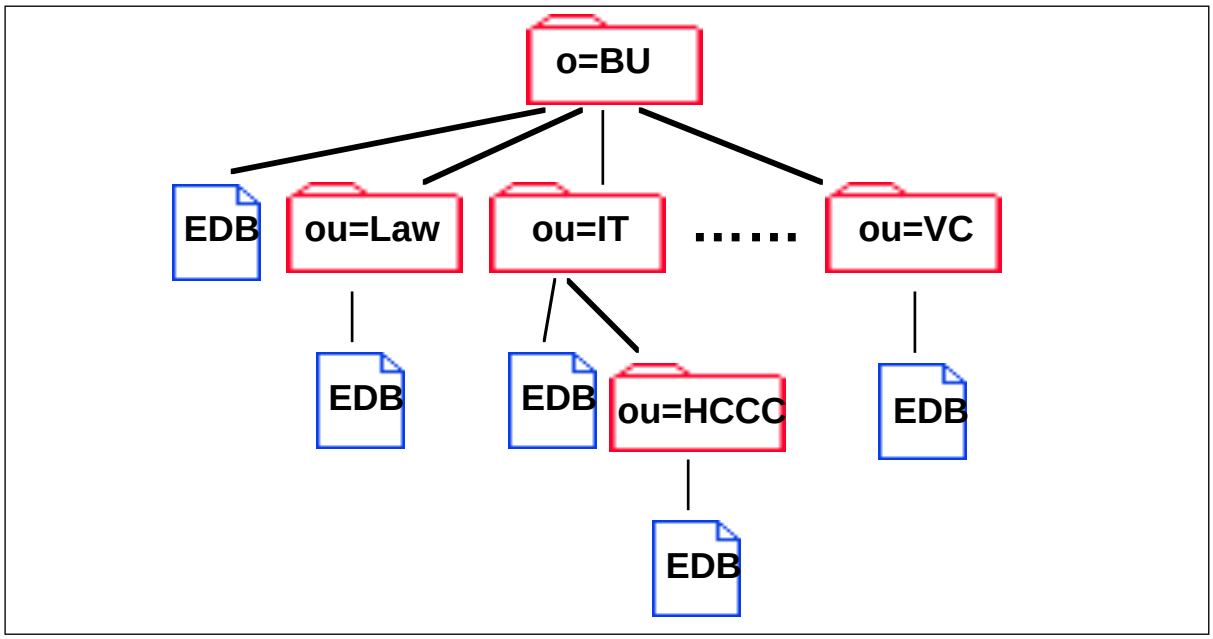


Figure 50: QUIPU Directory Structure Example

Future plans for QUIPU include:

- greater support for multiple database formats,
- support 1992 extensions,
- increased security and management, and
- conformance testing.

A complete discussion of QUIPU can be found in (Kille, 1991; Hardcastle-Kille, 1992c).

Lightweight Directory Access Protocol

The Lightweight Directory Access Protocol (LDAP) is a high-level API to the X.500 Directory Access Protocol (DAP). LDAP provides simple read, write and search interactive access to the directory. LDAP is designed to run over connection-oriented and reliable transports such as TCP/IP. LDAP is described in the Internet draft document (Yeong *et al*, 1992) with early work on similar protocols such as DIXIE (Howes *et al*, 1991).

The LDAP protocol uses a client/server model with clients transmitting protocol requests describing the operation to perform. The server, upon completion of the operation, returns a response containing any results or errors messages. The objective is to minimise the complexity of clients to achieve widespread deployment of applications utilising the directory.

LDAP provides synchronous and asynchronous routines for interrogating the directory. Table 19 shows an example of the source code required to instantiate a search of the directory with the result returned in the data structure 'SearchResult'. In this example, the main search routine arguments include the starting base of the search (portion of the DIT), the scope level of the search, a matching criteria filter, and the list of attributes to return. LDAP provides further routines to traverse and extract the Distinguished Names of found entries in the SearchResult

list including each attribute and their values. LDAP's simple API interface provides an effective mechanism to enable applications to interrogate directory facilities.

Table 19: LDAP Example Code

```
ldap_open( LdapServerName, Port )
ldap_bind( Ld, NULL, NULL, LDAP_AUTH_SIMPLE )
ldap_search_s( Ld, Base, Scope, Filter, SearchAttrs, ATTRSVALS, &SearchResult )
matches = ldap_count_entries( Ld, SearchResult )
```

4.7 Summary

This review has highlighted some areas on the graphical interfaces of common electronic mail applications that are consistent across the application domain. In particular these areas include:

- Message lists detailing the originator, date, subject and urgency level (where supported) of each message on a single line.
- Direct access and high visibility of message folders.
- Separate message composition windows with support for options.
- Message addressing with dynamic recipient lists and supporting recipient categories.

These high-level functions, although implemented differently on disparate platforms, are mandatory requirements for both X.400 and non-X.400 systems. A common formalisation of these functions would enhance the overall usability for the electronic mail systems domain. None of the systems reviewed considered the problems of large list interaction (eg message folders) and the problems of scaling to meet broader requirements.

Apart from the X.400 systems, there is no standard recipient address syntax. The other applications rely on third-party gateways for the interchange of messages. X.400 systems, with its standard O/R address format, suffers from a intricate syntax and forces the user to deal with complicated dialogue interfaces.

Little integration of X.500 Directory Services with current electronic mail systems has been realised. In fact, no non-X.400 systems utilise X.500 services. For most X.400 systems, the utilisation is usually provided by a separate X.500 DUA application to perform the lookup and requires the user to cut-and-paste the results into the message fields. The user-benefits of directory services will not be fully exploited until the closely coupled integration of directory services with messaging applications is realised. This process would then implicitly solve the non-standard addressing requirements for heterogeneous systems, through the common directory naming scheme.

Chapter 5

Design and Implementation

5.1 Introduction

This chapter outlines the design and implementation issues involved with the development of the Iris UA. The Iris Reference Model is based on the successful outcome of this work. Initially several development directions, both hardware and software, influenced the design strategy requirements. A number of UIMS were reviewed for this task and their suitability evaluated. The X.400 and X.500 aspects of the UA, particularly the IPM service elements, were investigated as the underlying support system for the prototype was necessary for realistic interaction.

The design strategy involved in the development of Iris is fully discussed. A 'user-centred' design philosophy was adopted, with the user's requirements and needs as the major driving force behind design decisions (Norman & Draper, 1986). The initial plan was to use a paper-based design followed by comprehensive use of a UIMS. The paper-based design would allow an early 'walk-through' evaluation and allow a greater understanding of the requirements for IPM UAs.

A discussion of the design and implementation of Iris using a UIMS is presented. Aspects and deficiencies of UIMS as prototyping tools are also highlighted. This led to changes to the overall design and implementation strategy and ultimate abandonment of early design work. The OSF/Motif widget library was utilised for subsequent implementation.

The various methods of X.500 directory integration for X.400 message addressing are comprehensively discussed. A new algorithm, utilising the LDAP system, is presented which attempts to maximise the search and matching success rate for directory lookups. The algorithm adds quasi-intelligence to the directory searching interface and follows a wide range of searching paths. Message addressing using O/R addresses is also considered as it presents a common situation for X.400 message submission.

The design of the Iris UA follows de facto standards for this class of application to present a consistent and usable interface. Some aspects are purposely different since X.400 UA interfac-

es are a new breed of electronic mail systems and require novel and user-empowering interfaces and interaction methods.

Throughout this chapter references are made to the user interface guideline series developed by Smith & Mosier (1986) that were used in the design of the Iris UA. The guidelines assisted in the development process by providing a set of criteria to base design decisions upon. The use of automated hypertext software (Iannella, 1992a) to search and gather related guidelines expedited this task, particularly for large user interface guideline sets. The Smith & Mosier guidelines are quoted as [SAM-9.9/99] where the latter numbers are the guideline section and number. Each guideline quoted is listed in Appendix E with the full name and guideline statement.

It is important to note that a large subset of the Smith & Mosier guidelines are implicitly satisfied in the Iris UA since the OSF/Motif user interface was developed with comparative attention to existing guidelines. The guidelines quoted in this chapter are primarily targeted towards electronic mail systems, form-based screens, and large list management. Even so, of the 83 user interface guidelines devoted to Data Transmission, almost half of these criteria are also implied as they are provided by the underlying X.400 and X.500 layers.

5.2 Implementation Issues

A number of implementation issues arose during the development of Iris that had substantial influence on the direction taken in the design methodology. A number of UIMS and interface toolkits were evaluated for their suitability for the development. The underlying X.400 and X.500 systems were also investigated.

When the decision to use the X Window System as the basis for the development of Iris was made, the only interface toolkit available was the Athena[1] widget set. This primitive toolkit was not sufficient and the HP[2] widget set was investigated. The HP widget set, although richer, was not suitable in the long term as it was dependent on one vendor. Subsequently, the OSF/Motif[3] widget set was announced. This provided the necessary interface objects for contemporary graphical X workstations. (Interestingly, the OSF/Motif widget set was heavily based on the HP widget set.) Recent industry announcements[4] have confirmed the OSF/Motif graphical user environment as the de facto standard.

The evaluation of development tools revealed a number of public domain X Window System toolkits, user interface languages, and UIMS. A few early systems reviewed did not provide support for the OSF/Motif widget set. These include:

- DIRT[5] (Design In Real Time)
- Tcl/Tk[6] (Tool Command Language/ToolKit)
- Garnet[7]

1. Athena widget set is a trademark of the MIT X Consortium.
2. HP widget set is a trademark of the Hewlett-Packard Company.
3. OSF/Motif widget set is a trademark of the Open Software Foundation.
4. Common Open Systems Environment (COSE) announcement on June 8, 1993.
5. DIRT was developed by Richard Hesketh, Computing Laboratories, University of Kent at Canterbury.
6. Tcl/Tk was developed by John Ousterhout, University of California at Berkeley.
7. Garnet was developed by Brad A Myers, School of Computer Science, Carnegie Mellon University.

The systems reviewed that did support the OSF/Motif widget set included:

- Wcl[8] (Widget Creation Library)
- Serpent[9]
- WINTERP[10] (Widget INTERPreter)

A major deficiency of all these systems was that they all define their own user interface language to describe the widget objects. Each system had a meta-language that described the interface and hence, would be incompatible with any other system. This would not be a sensible development decision as there would be serious portability constraints.

The final decision was to use a commercial UIMS called XBUILD[11]. Along with all the other advantages of a graphical UIMS (see '2.6 User Interface Management Systems' on page 30), XBUILD could also export the interface as standard C code, or User Interface Language (UIL). This advantage was seen to be very important in the long term for the project.

Another issue that arose during development was the advantage of using object-oriented programming techniques. Since the OSF/Motif widget set is not object-oriented, a number of external systems have been developed that add this layer. These include:

- WWL[12] (Widget Wrapper Library for C++)
- Motif++[13]
- GINA++[14] (Generic Interactive Application for C++ and OSF/Motif)
- Xm++[15]

These systems also have one major deficiency; they each encapsulate the OSF/Motif widget set in a non-portable and system-dependent language. It was obvious that until the X Window System is fully object-oriented, the high level widget sets would not be able to be encapsulated without system dependencies and hence, portability issues. However, the underlying data structures (eg messages) used in the Iris UA were developed using the C++ object-oriented language.

The decision as to which X.400 and X.500 systems to utilise was considerably less involved. The PP and QUIPU systems were the obvious choice as they were freely available and provided all the standard services. The use of LDAP (and its predecessor DIXIE) for access to the X.500 services was also exploited. A early look at Digital Equipment Corporation's 'VAX Message Router X.400' proved fruitless as it was a layered product with no integration opportunities. Also, a review of the use of Omnicom's X.400 and X.500 APIs, although promising, were not considered as the person-years involved in developing implementations from the specifications was overwhelming.

---

8. Wcl was developed by David E Smyth, Jet Propulsion Labs, California Institute of Technology
9. Serpent was developed at the Software Engineering Institute, Carnegie Mellon University.
10. WINTERP was developed by Niels P Mayer *et al*, Hewlett-Packard Laboratories.
11. XBUILD is a registered trademark of Siemens Nixdorf Information Systems Inc.
12. WWL was developed by Jean-Daniel Fekete, Laboratoire de Recherche en Informatique, Faculte d'Orsay, France.
13. Motif++ was developed by Ronald Van Loon, University of Lowell, Netherlands.
14. GINA++ was developed by Andreas Backer & Andreas Genau, German National Research Centre for Computer Science.
15. Xm++ was developed by Bernhard Strassl, University of Vienna.

Finally, a number of interesting technical issues arose that were related to the hardware development environment. The environment moved from DEC's VAX Ultrix, to Apple's A/UX, then finally to DEC's MIPS Ultrix. Each move required the establishment of new installations of the various support layers. These included the X Window System, OSF/Motif, ISODE, PP, QUIPU, DIXIE, and LDAP. Figure 51 shows the relationship of the various implementation layers involved in the development.

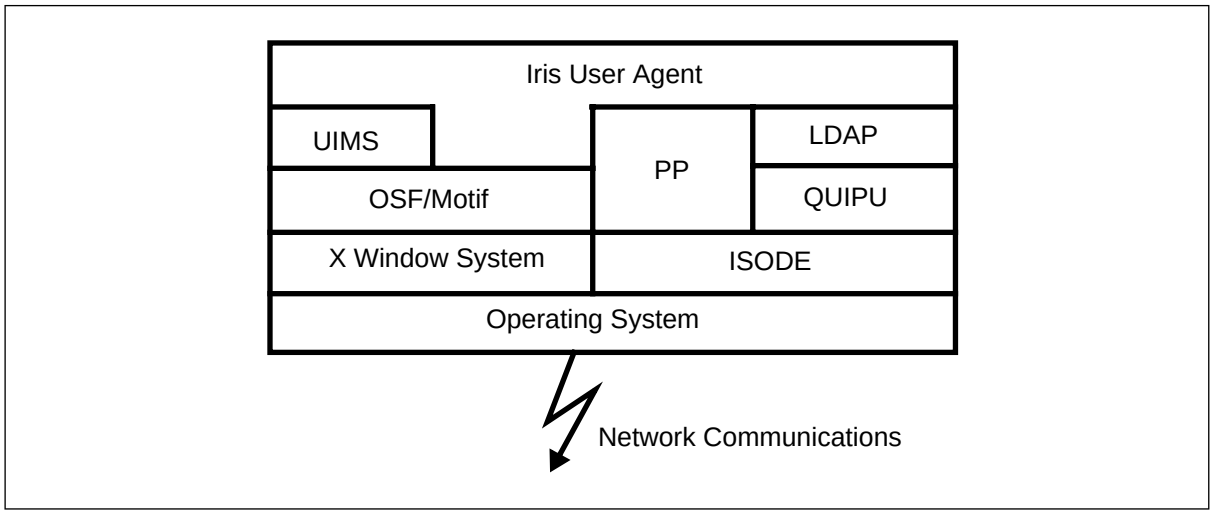


Figure 51: Iris Development Layers

5.3 Design—Stage One

The first stage of design of Iris was to develop a series of paper-based screens based on the X.400 standard. This involved a comprehensive study of the X.400 series, and in particular, the X.420 standard on IPMs. The paper-based screens could then be evaluated using three methods:

- An initial walk-through evaluation.
- Conformance to Smith & Mosier guidelines.
- Applicability of usability heuristics.

Designing an interface for an X.400 UA is a formidable task as both the standard is complex and the UA interface is not defined. Exploiting a graphical user interface can facilitate the many screens and options required to provide all the UA services. Rees & Iannella (1990) describe design strategies for graphical user interfaces to electronic mail systems that lead to convenient and effective interfaces to X.400 systems.

The initial design was based on the structure chart derived from the standard (see Figure 52) which highlights four natural sections of X.400 MHS:

1 authentication,
2 origination,
3 reception, and
4 management.

These sections are captured in the opening screen of Iris (see Figure 53). (Note: The initial working title of Iris was X4Mail.) The authentication stage is assumed to have been completed when the user established connection to the workstation. The UA indicates this by displaying

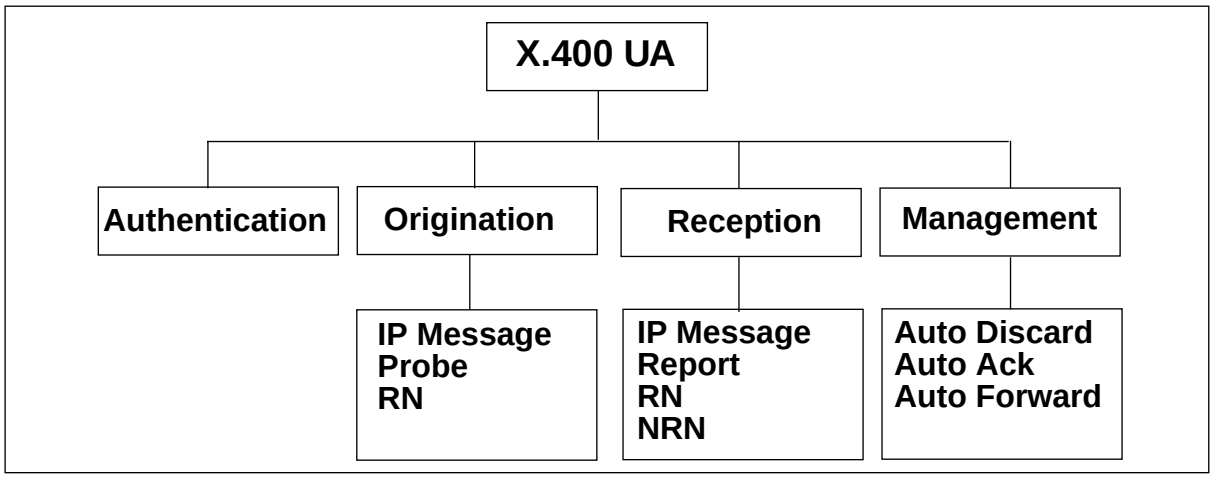


Figure 52: UA Design Structure Chart

the O/R address of the user as feedback. The UA design can now concentrate on the other three sections.

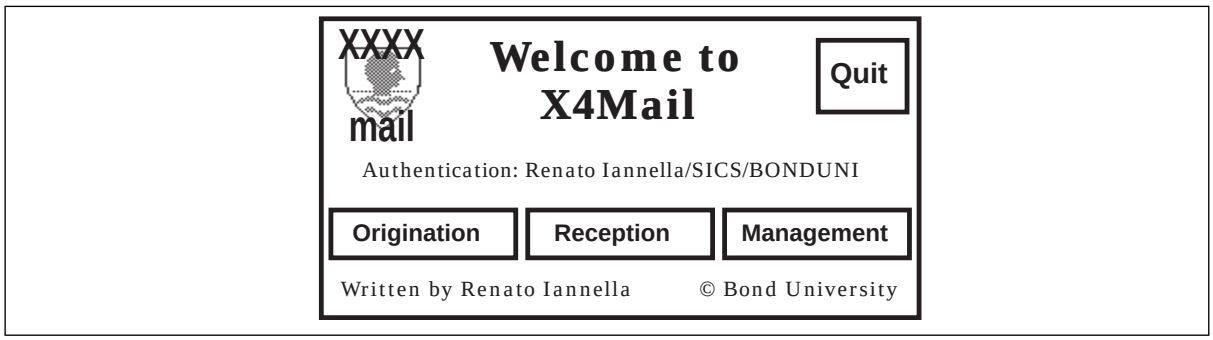


Figure 53: Stage 1—X4Mail Main Screen

The design of the origination screen (see Figure 54) incorporates a list of draft messages and a series of buttons and text fields enabling an IPM to be created. The list of draft messages are IPMs that have been partially composed but not sent. If any of the draft messages are selected, then the header and body fields would be displayed. The subject and body part are text fields with standard editing facilities. The recipients and options buttons generate new dialogs and also show which options and recipients have been selected in the scrolling list adjacent to each button. The bottom of the screen shows a series of buttons to compose an new IPM, save the current IPM as a draft, remove the IPM, or send the IPM.

The recipients button displays a new dialog in which the user may select from the four different categories of recipients (see Figure 55). The screen shows a list of previously defined nicknames to select from and a button to create new nicknames. The design was created to utilise the ability to drag the nicknames to the four scrolling lists of recipients [SAM-5.2/6].

The design of the nickname screen for the creation of the full O/R address was a non-trivial process. The O/R address may be supplied as a single text string or a series of attribute/values pairs. In any case the user must be aware of which parts of the O/R address are attributes and which are the corresponding values. The most appropriate alternative is to present the user with a form screen [SAM-3.1.2/1] to be filled in with the components of the recipient's O/R address (see Figure 56). Such form-filling screens mimic the paper version in an attempt to provide a consistent interface [SAM-1.4/24, SAM-1.4/25]. This screen is seen to be of critical

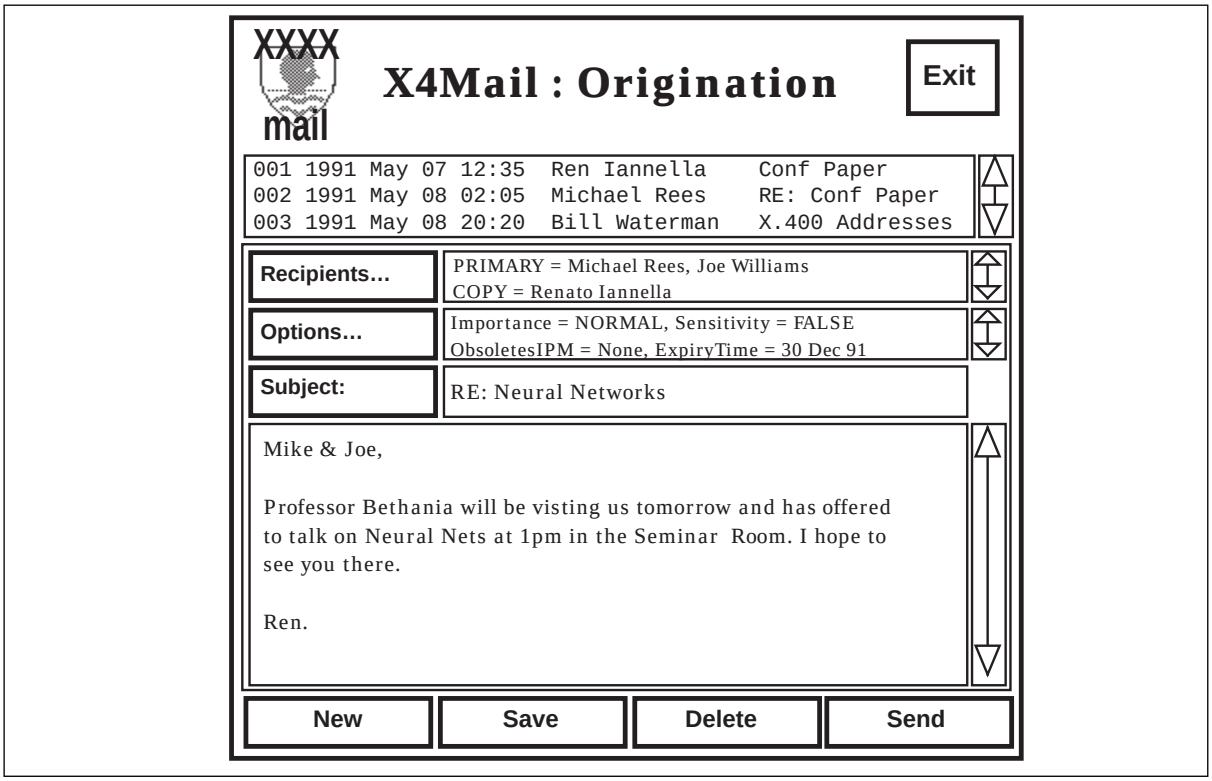


Figure 54: Stage 1—X4Mail Origination Screen

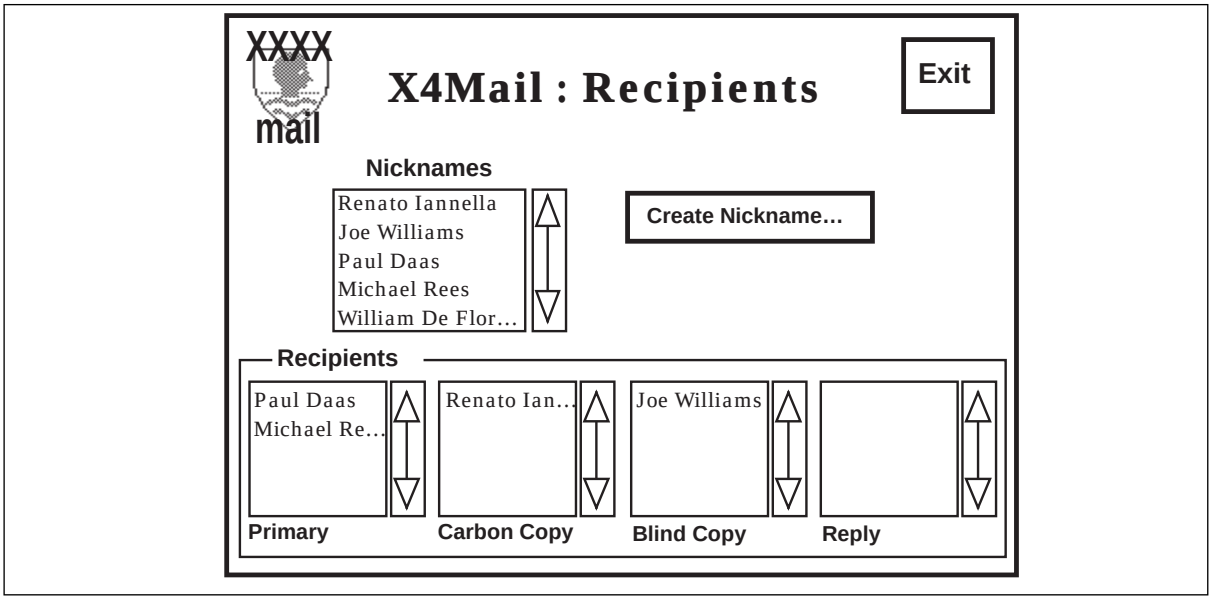


Figure 55: Stage 1—X4Mail Recipients Screen

importance to the UA as incorrect specification of the recipient's O/R address, due to the interface, will result in non-deliverable messages.

The design of the message options screen involved a series of radio buttons and text fields to specify the range of IPM optional service elements. The radio buttons include options to specify the importance [SAM-5.3/7] and range of sensitivities for the message as well as indicating if the message was to be treated as a probe (ie to validate the deliverability of the message). Two text entry fields allow the user to specify any related IPMs or any IPMs that the current

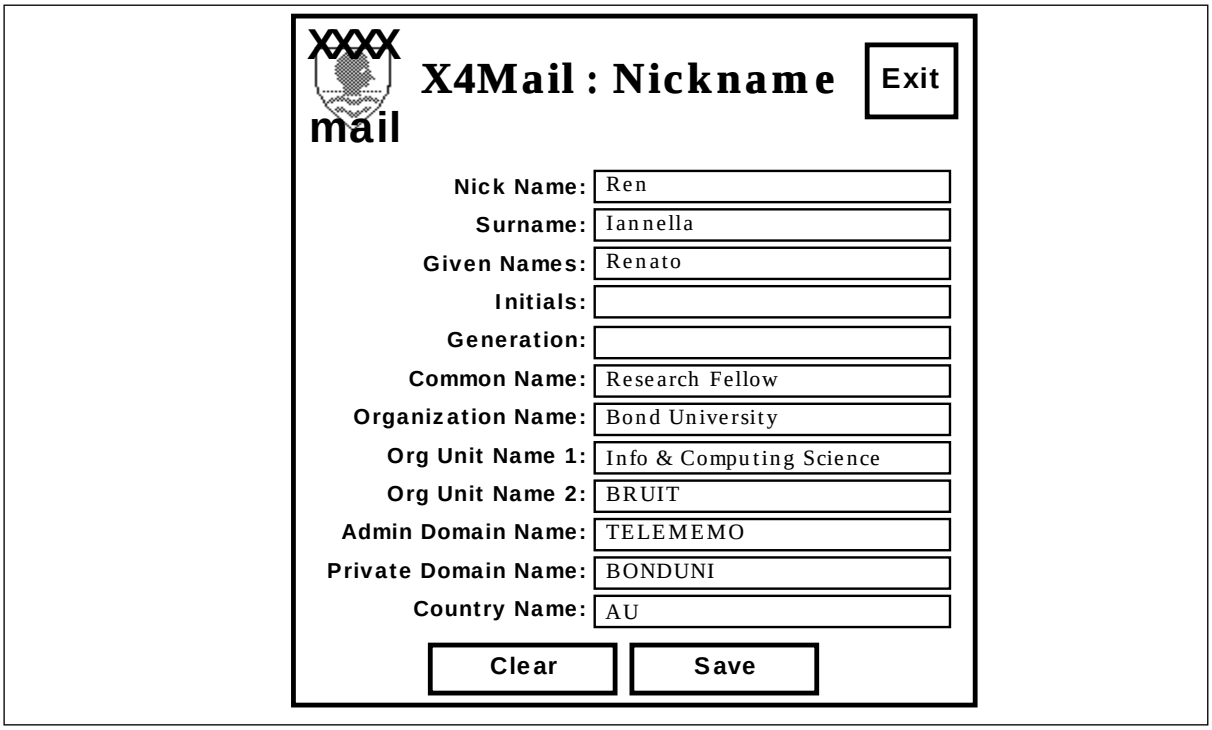


Figure 56: Stage 1—X4Mail O/R Address Screen

IPM will obsolete. In this case, the IPM identifier must be entered. The message expiry and reply times are also entered via two text fields. Figure 57 shows the message options screen.

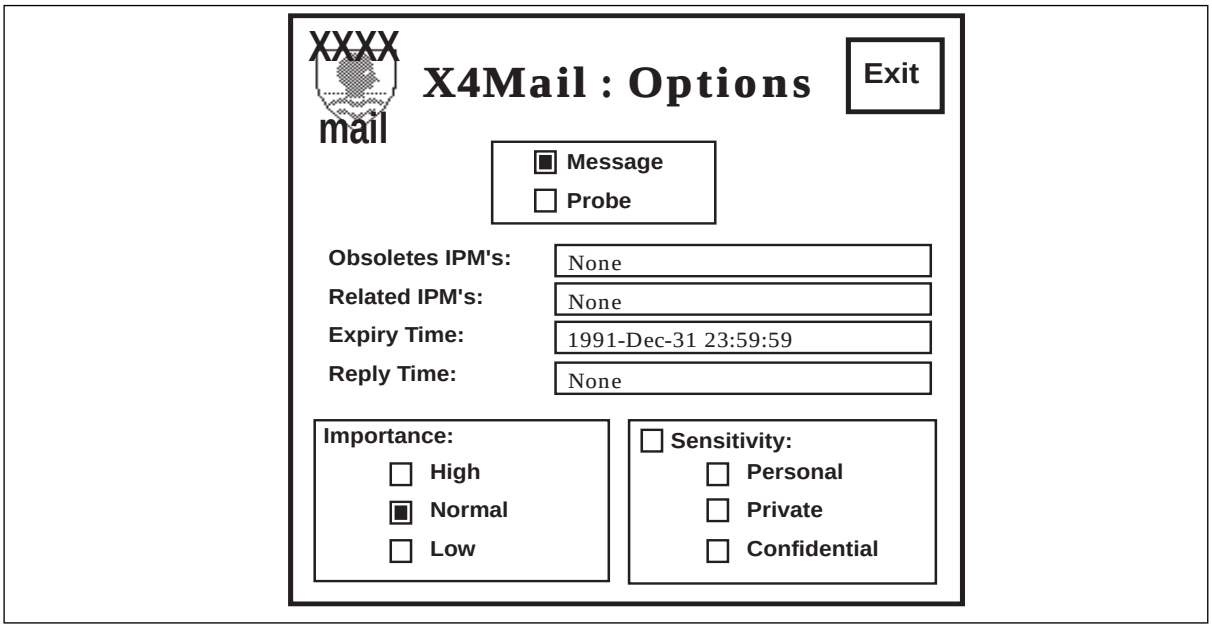


Figure 57: Stage 1—X4Mail Options Screen

The design of the management screen includes check boxes and text fields to specify the appropriate management facilities (see Figure 58) [SAM-5.4/3]. Check boxes are used to indicate the automatic discarding of expired and obsoleted IPMs as well as the automatic acknowledgement of new IPMs (ie automatically generate a RN). Automatic forwarding of IPMs is available via a check box and a series of text fields allows the specification of the forwarding O/R address and a heading and comment field.

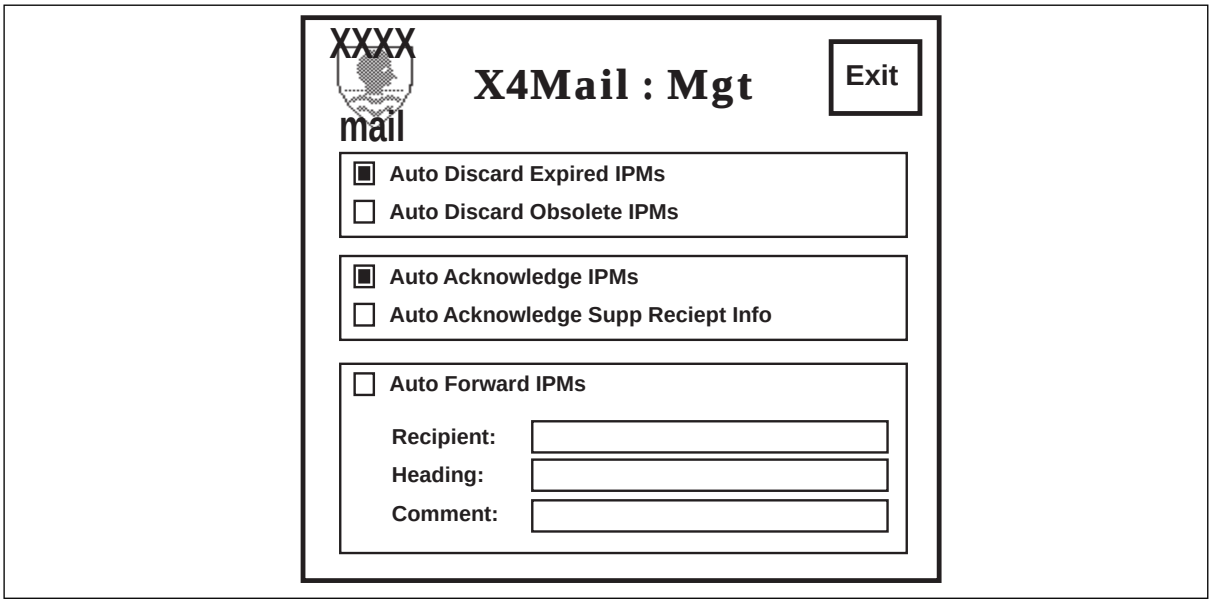


Figure 58: Stage 1—X4Mail Management Screen

5.3.1 Evaluation and Redesign

At this point, a review of all the designs was warranted to enforce user interface consistency and to plan screens still to be designed (eg the reception window had not yet been considered). Also, a further review of the IPM X.420 standard revealed some omissions and incorrect interpretations in the initial designs:

- A number of message options had been omitted.
- The auto-acknowledgement option 'Supp Receipt Info' was incorrectly designed as a check box when it requires a text string to be entered.
- The management of message folders was required for archiving and retrieving messages.
- The support for nickname modifications was also required to manage the O/R addresses.

In most cases, the design of the screens led to a greater understanding of X.420 as it was intricately dissected to determine all possible options and requirements for IPM UAs. Some new options were highlighted (eg deferred delivery) and some were removed (eg probes and manual RN generation) as their utility was seen to be limited.

A simple walk-through evaluation of the first set of designs emphasised some screen layout issues, in particular, the layout of the screen interface objects, and the separation and grouping of similar objects. Some options were found to be tedious for user entry and new options were added. Easier access and specification of some options was highlighted and addressed with new interface objects. Also, applying usability heuristics (such as Molich & Nielsen, 1990) provided extra awareness into the early identification of user interface problems. The overall screen map for the new window layout is show in Figure 74.

Figure 60 shows the new main Iris window. The main change was the renaming of the three X.400 sections to Submission (from origination), Correspondence (from reception), and Administration (from management) in an attempt to further move away from X.400 terminology when not required [SAM-5.0/2]. Also, the new names closely reflected their functionality. The other small change was with the authentication field which now displays the user's O/R ad-

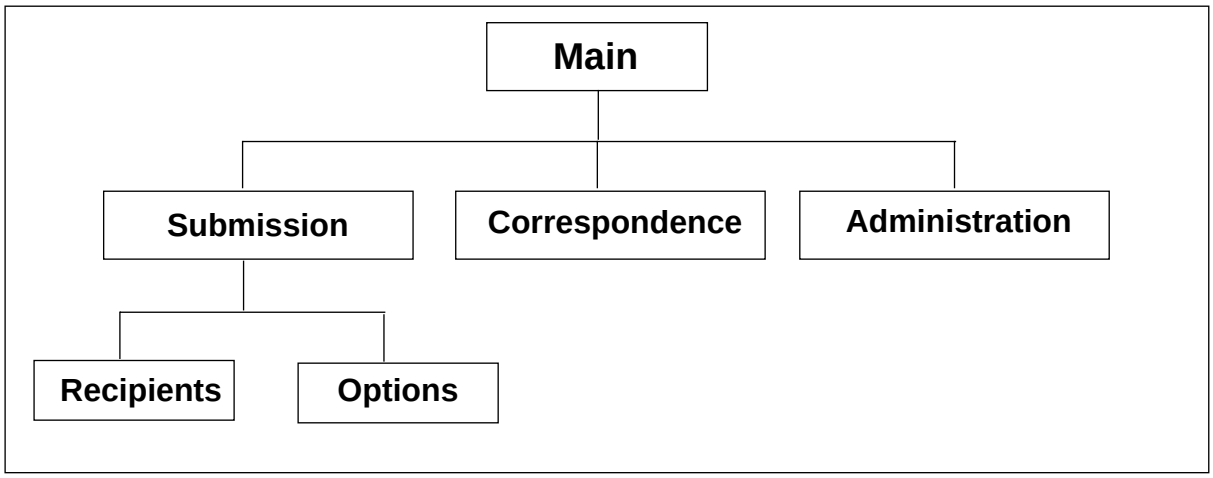


Figure 59: Stage 1—Iris Screen Map

dress in the standard notation. Two buttons, Help and Quit, were also added to the main Iris window.

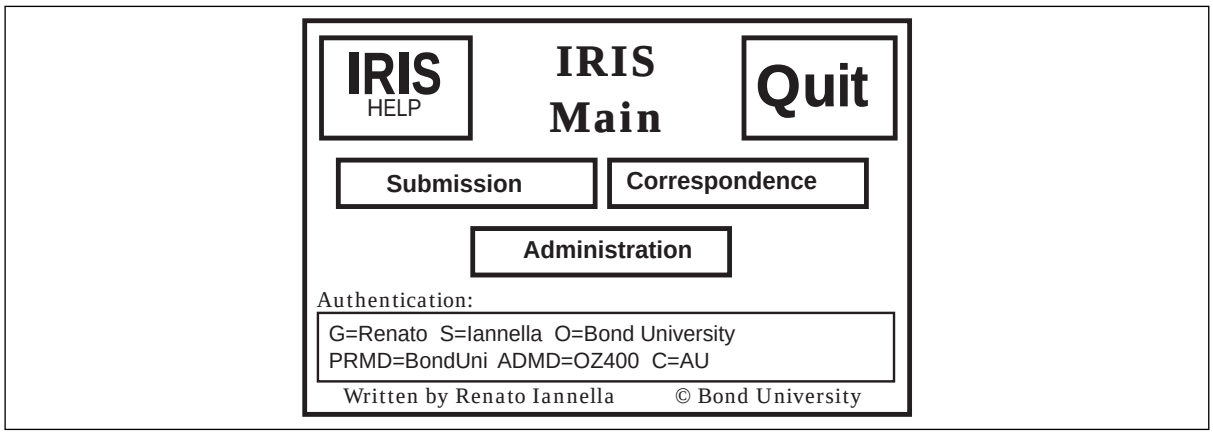


Figure 60: Stage 1—Iris Main Window

Apart from some layout changes, the submission window's only major changes was the removal of the list of draft messages and the New button (see Figure 61). The draft messages now became a part of the correspondence window. The current message can be saved as a draft for later completion [SAM-5.1/13]. The New button is no longer required since the Submission button (from the main window) performs the same function.

Similarly, the layout of the message recipient's window only major change being the relocation of the Create Nickname button to the administration window (see Figure 62) This would enable the user to modify existing O/R addresses.

The message options window (see Figure 62) now includes two option menus for the specification of obsoleted and related messages [SAM-5.0/10]. The option menus would display a list of IPM identifiers allowing the user to select the correct value easily and more effectively. This list may become very long as more messages are stored by the user and an option menu may not be the correct solution in the long term.

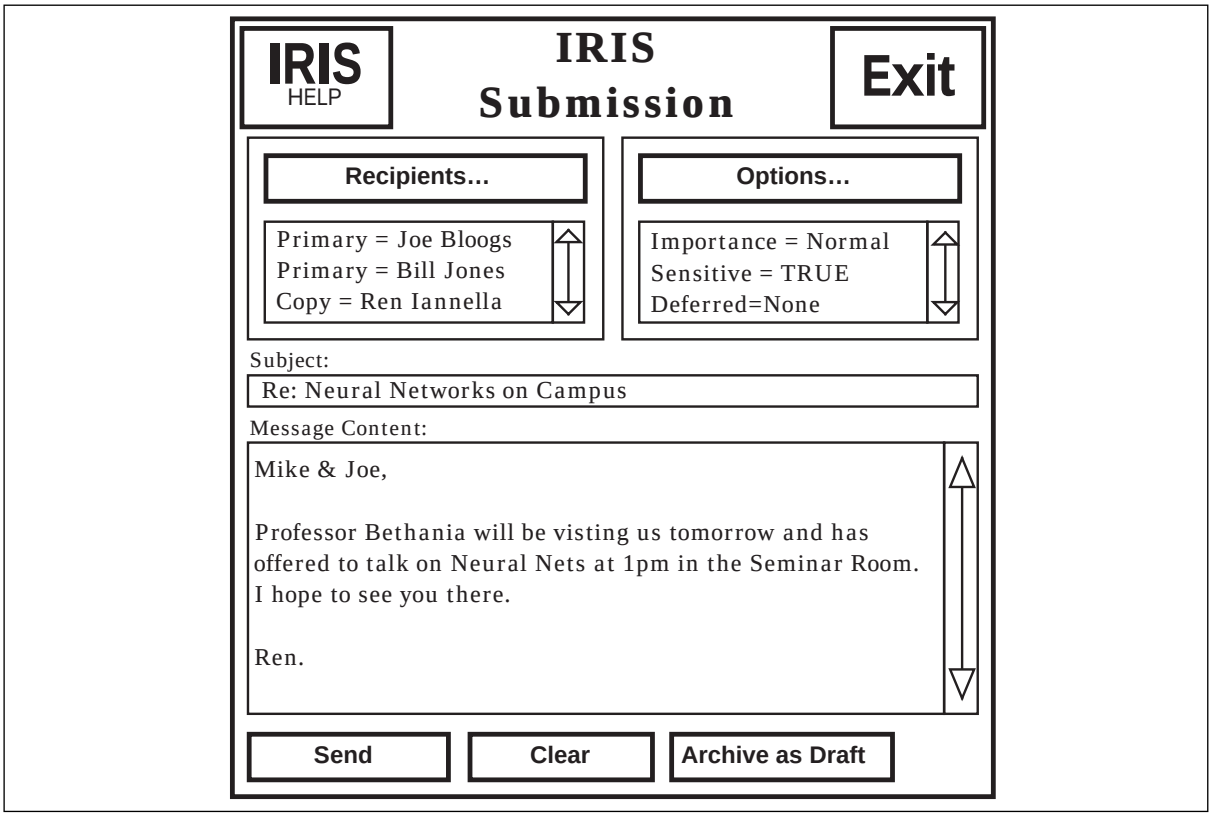


Figure 61: Stage 1—Iris Submission Window

The authorising user[16] of the IPM has been added and also includes an option menu of the predefined nicknames to select from. The message expiry and reply times have been modified to allow for option menu selection of the relevant times. The use of option menus to provide input mechanisms significantly improves the specification of IPM service elements [SAM-5.0/4]. The validity of the selections is also greatly improved for both the user and developer.

The design of the correspondence screen includes mechanisms to retrieve and display previously archived messages and draft messages. The screen includes a list of archive folders and a corresponding list of messages contained in the currently selected folder. The message directory has the familiar one-line summary of the messages and an indication of the current folder being viewed [SAM-5.0/3]. When selected, the envelope and content of the message can then be displayed. The buttons on the bottom of the screen allow the user to reply, forward, or remove the message and to check for any new mail that has arrived. The UA would also automatically check for new mail arrival after a fixed time interval and notify the user in a non-disruptive manner [SAM-5.5/5].

The administration screen was redesigned to include a number of new facilities and improve on the existing ones (see Figure 64). The receipt notification now includes a Supplementary Information text field and the automatic forwarding of messages now utilises an option menu to specify the nickname of the recipient. Three new facilities were added to manage Archive Folders, Nicknames, and Deferred Delivery messages. Both the Archive Folders and Nick-

16. This option indicates the person(s) who authorised the message, not the author (originator) of the message. See Table 8, 'IPM Heading Fields', on page 65.

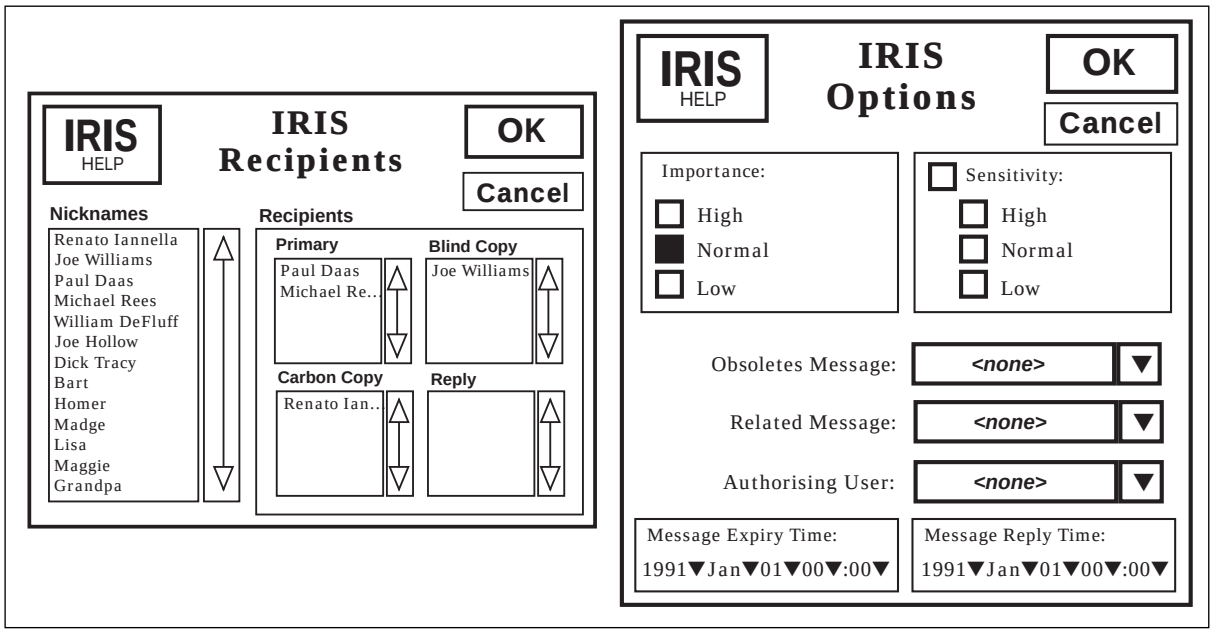


Figure 62: Stage 1—Iris Recipients and Message Options Window

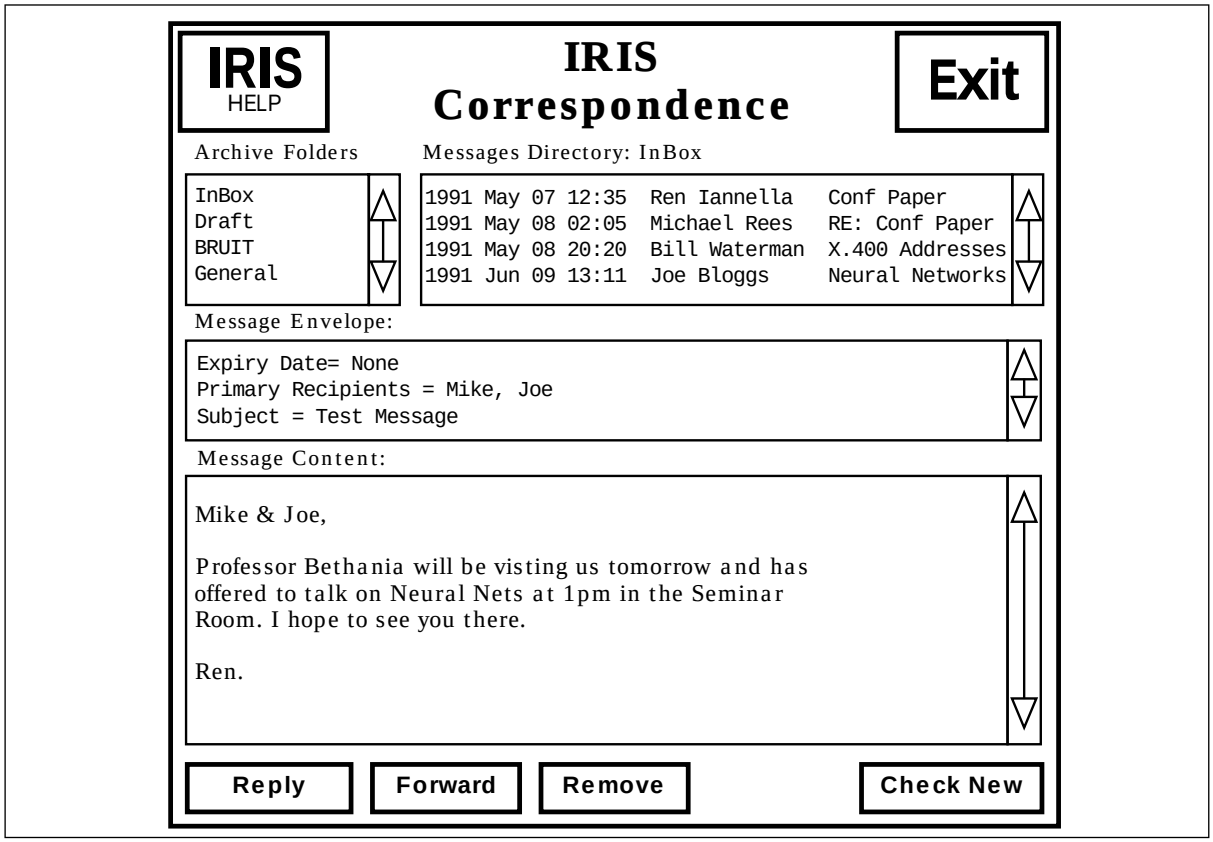


Figure 63: Stage 1—Iris Correspondence Window

name management allow for the creation, removal, and specification of new folders for message storing and new O/R addresses [SAM-5.2/7].

The Deferred Delivery directory lists current messages that are currently awaiting delivery [SAM-5.3/9]. The deferred delivery time would be specified in the same manner as the expiry and reply times in the message options screen (see Figure 62) [SAM-5.3/10]. Two other options allow the deferred messages to be removed or put on hold indefinitely.

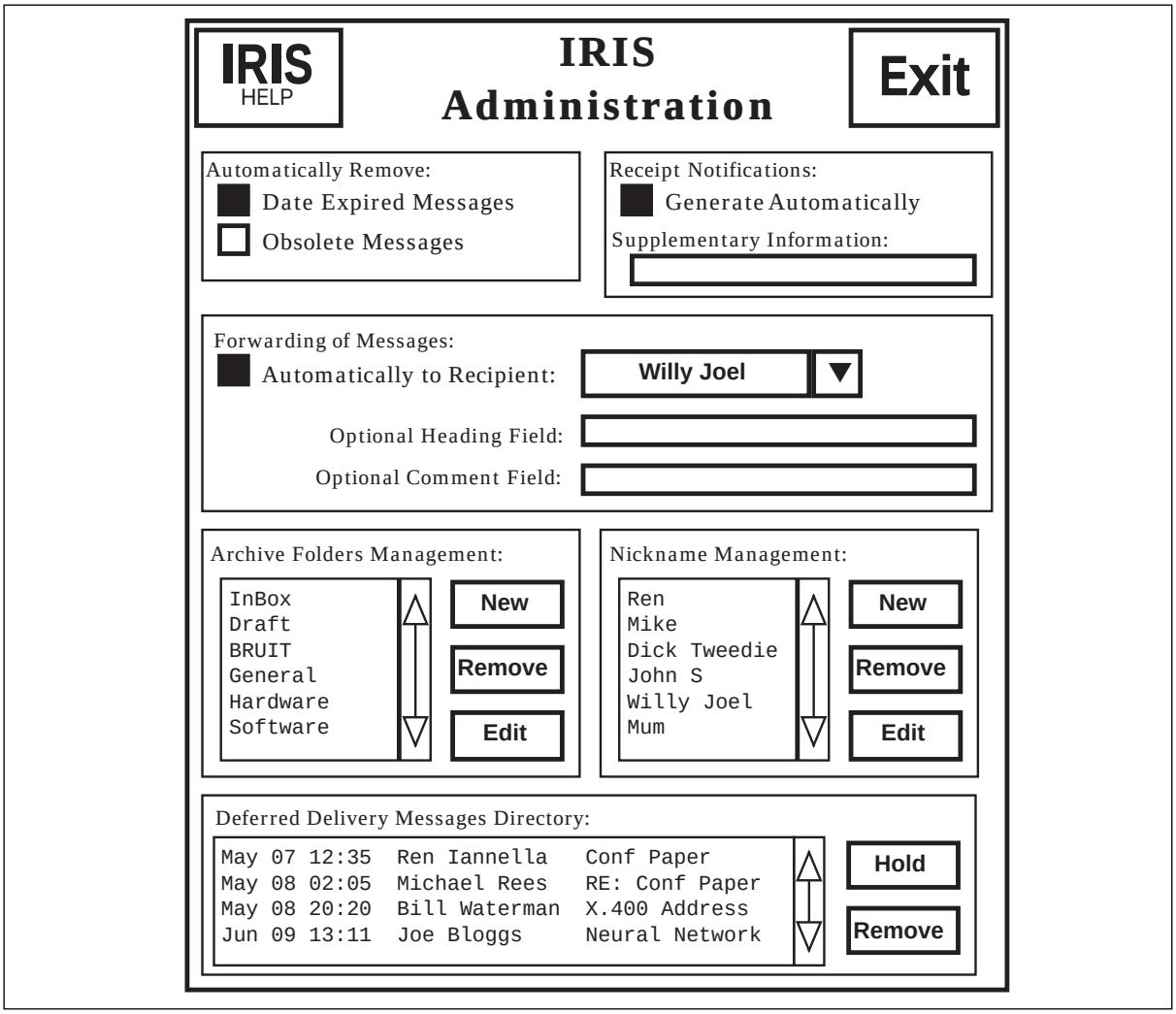


Figure 64: Stage 1—Iris Administration Screen

At the end of the paper-design stage, a further walk-through evaluation was performed and consultation with the X.400 standard concluded that an appropriate level of functionality had been achieved. The benefits included:

- Direct access to the submission, correspondence, and administration functions.
- Message submission with complete option specification of IPM service elements.
- Support for message recipient categories.
- Message recipient specification via nicknames.
- Complete message archiving and retrieval with folder management.
- Full administrative options with deferred delivery management.

These benefits were consistent with many guidelines [SAM-5.0/6, SAM-5.0/7, SAM-5.1/1, SAM5.3/2] and compatible with existing messaging applications. The next stage of the design process now required the transformation of the paper-based screens into a demonstrable prototype.

5.4 Design—Stage Two

Stage two of the design and implementation of Iris involved development with the XBUILD UIMS. XBUILD is a graphical UIMS allowing OSF/Motif widgets to be directly manipulated

to develop the layout of the interface. After some initial training, XBUILD was used to transform the designs from stage one into a working prototype. Although this process was reasonably straightforward, XBUILD (and other OSF/Motif UIMS applications) did require intimate knowledge of the widget hierarchy in order to develop the screens.

One of the first design decisions was to use OSF/Motif paned windows to separate the interface objects on each screen. This would provide a consistent look and mechanisms for the user to manipulate screen sizes [SAM-2.0/6]. The top panel of each paned window includes the title, a help button, and (if appropriate) a close button. The bottom panel includes a list of functional control buttons (such as OK and Cancel). Heavy use of the constraint-based form widget was also utilised as the manager for all other interface objects. When properly set up, the form widget allows each window or pane to be resized and appropriate scaling is performed on the managed children widgets.

The opening main window of Iris simply consisted of a title pane and four pushbuttons for each of Submission, Correspondence, Administration, and user Preferences. The correspondence window incorporates a scrolling list of archive folders, with pushbuttons to manage the folders, and a scrolling list of the messages directory. The management of the archive folders has now moved to this screen from the Administration screen developed in the first design stages (see 'Figure 64: Stage 1—Iris Administration Screen' on page 119). The main reason for this change is that the user now has direct access to this facility and is not required to navigate to the administration screen. The management of archive folders is of such a common occurrence for the user to warrant this change.

Each message envelope and content is displayed in scrolling text fields on the lower half of the correspondence window. The bottom pane of the correspondence window includes control buttons to reply, forward, and remove messages, and to check for newly arrived messages. Figure 65 shows a screen image of the correspondence window.

The submission window consists of a window title and two scrolling lists showing the list of message recipients and state of the message options. Two pushbuttons (Recipients and Options) above each list allowed the specification of recipients [SAM-5.2/1] and options [SAM-5.2/16] via additional dialog windows. The message subject constituted a small one-line scrolling text field and the content was a larger multi-line scrolling text field [SAM-5.1/3, SAM-5.1/12]. The bottom pane includes pushbuttons for sending or clearing messages, and archiving the message to the drafts folder. Figure 66 shows a screen image of the submission window.

The message recipients window consists of a window title and a large scrolling list of nicknames with pushbuttons to add, edit, or remove nicknames. The management of the nicknames has now also moved to this screen from the Administration screen developed in the first design stages (see 'Figure 64: Stage 1—Iris Administration Screen' on page 119). As with the archive folders, the main reason for this change is that the user now has direct access to this facility and is not required to navigate to the administration screen. The user would require quick and frequent access to the nickname management facilities on a regular basis as it is not uncommon to add and change O/R addresses during message preparation. Some systems that do not allow these direct enhancements, such as Pine (Siebel *et al*, 1992), reduce the user's ability for frequent changes to this information.

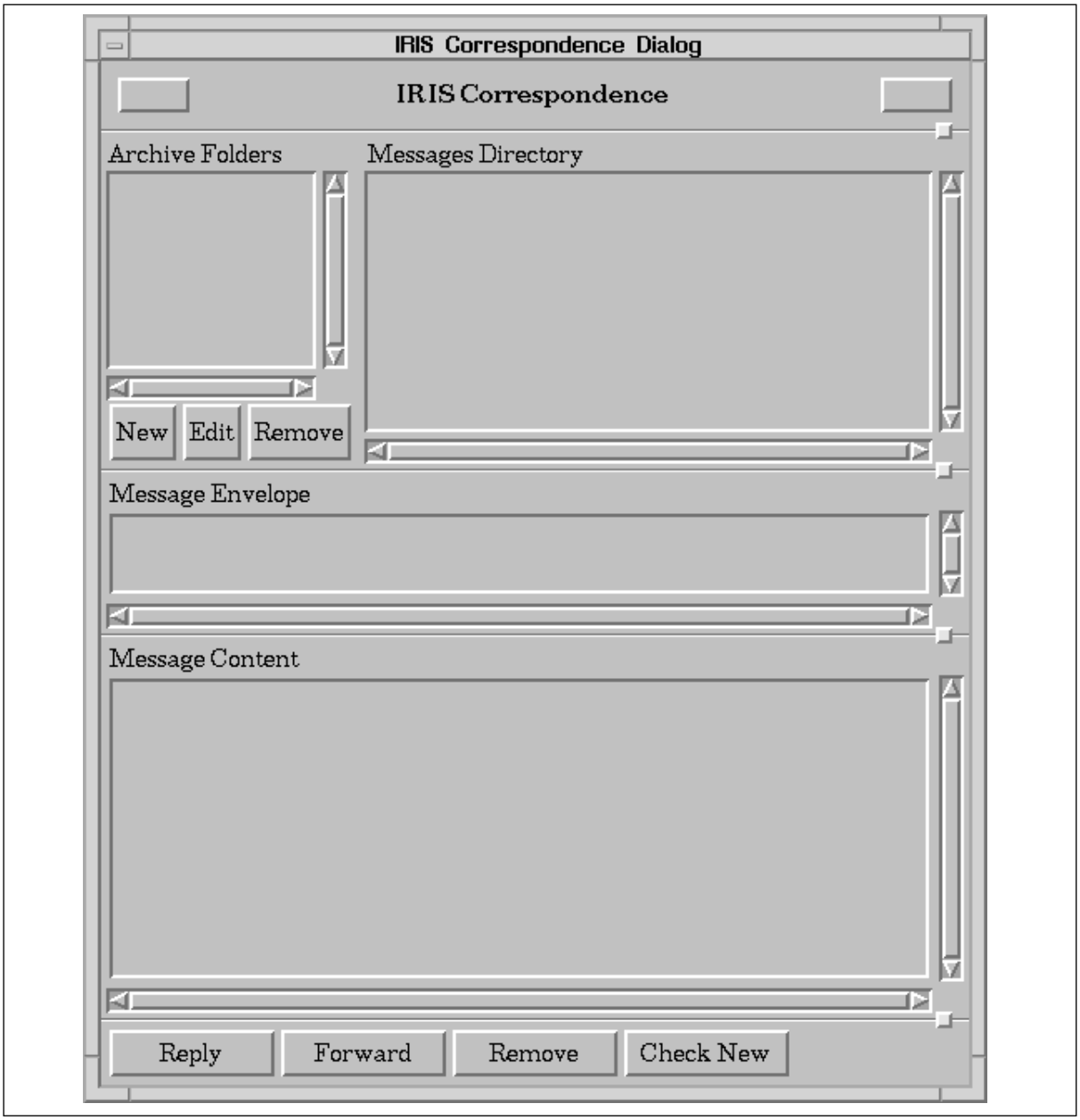


Figure 65: Stage 2—Iris Correspondence Window

Next to the nickname list are four smaller scrolling lists for each of the four recipient categories:

1 Primary,
2 Carbon Copy,
3 Blind Copy, and
4 Reply.

With this screen layout, the user would be able to 'drag' names from the nickname list and 'drop' them onto the appropriate recipient category list [SAM-5.2/2, SAM5.2/3]. The bottom panel includes Pushbuttons for OK and Cancel. Figure 67 shows a screen image of the message recipients window.

The message options window consists of a window title and two sets of toggle buttons to indicate the Importance and Sensitivity of the message. The Importance toggle buttons include;

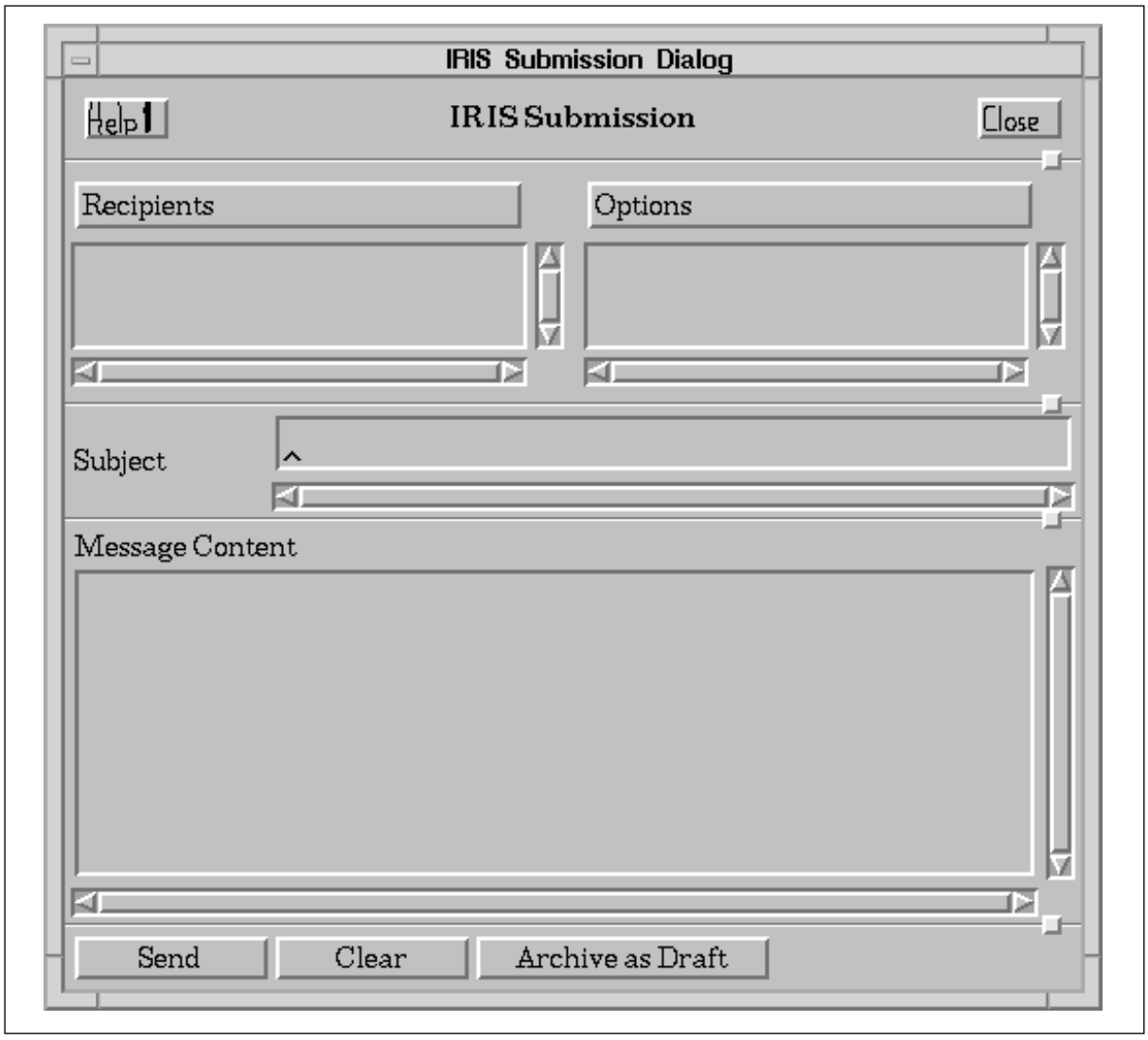


Figure 66: Stage 2—Iris Submission Window

High, Normal (the default), and Low. The Sensitivity toggle buttons include; None (the default), Personal, Private, and Company-confidential. Note that due to some layout difficulties with XBUILD, the two-level treatment of the sensitivity options from the first design stages (see 'Figure 62: Stage 1—Iris Recipients and Message Options Window' on page 118) was changed to a single level. However, one advantage of this layout is that it is now consistent with the importance options.

The next pane of the message options window allowed for the entry of the message's authorising user in a text field. Either a predefined nickname of a full O/R address could be entered. The next pane consisted of two more text fields for the entry of the message expiry and reply-by times. The bottom pane includes pushbuttons for OK and Cancel. Figure 68 shows a screen image of the message options window.

The nickname options window consists of a window title and a series of text fields for the entry of the O/R address attributes. An option menu was also available for the selection of the Country attribute as this was a finite list. The next pane consisted of a pushbutton to perform an X.500 search on the recipient details entered. The bottom pane includes pushbuttons for OK and Cancel. Figure 69 shows a screen image of the nickname options window. (Note: The

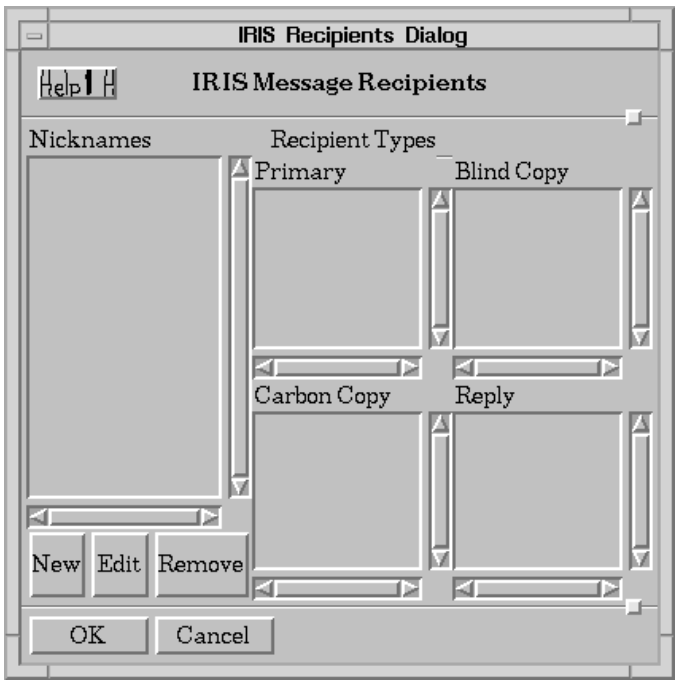


Figure 67: Stage 2—Iris Message Recipients Window

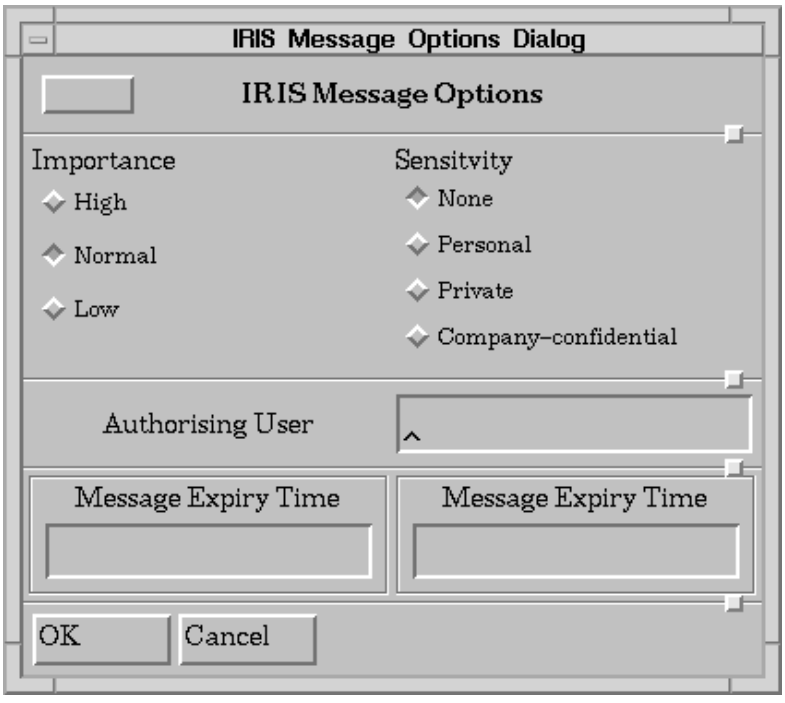


Figure 68: Stage 2—Iris Message Options Window

Country option menu is partially obscured due to some more technical difficulties with XBUILD.)

The XBUILD UIMS also allowed for a test-mode that enabled the interface to be simulated in real-time. The main importance of XBUILD was the generation of C language code for the screens developed. This allowed the prototype to be generated and easily compiled to a stand-alone application. XBUILD also supported the generation of callback stubs for each in-

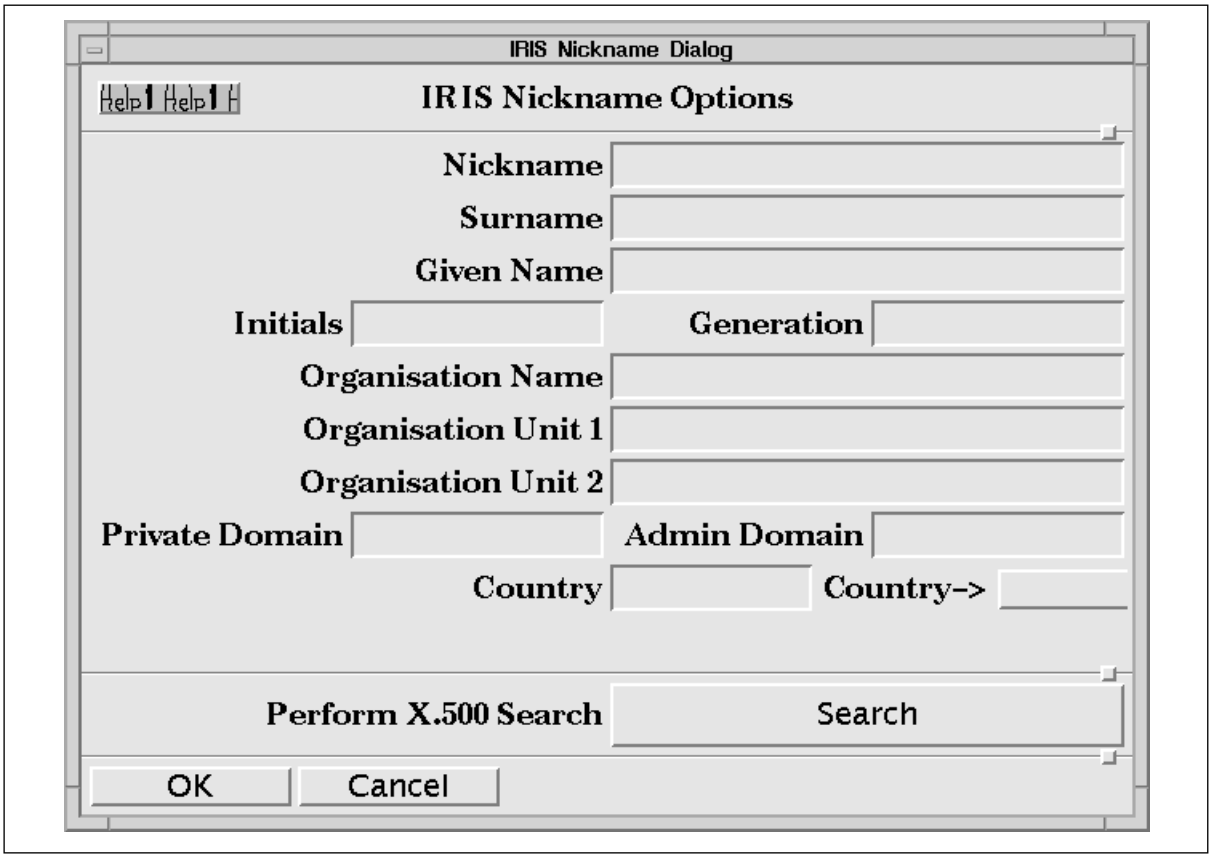


Figure 69: Stage 2—Iris Nickname Options Window

terface widget. These callback routines were executed whenever an appropriate event occurred such as a pushbutton being selected.

Unfortunately, XBUILD proved to be a very unreliable development environment with continual application crashes with destructive consequences. The software manufacturer was unable to offer any solutions to the problems and, eventually, XBUILD was withdrawn from the market.

It was also becoming clear that a number of requirements of the prototype may not have been possible with the current functionality of XBUILD. These include:

- Functions to create dynamic option menus were lacking for such objects as the time specification fields and nickname lists.
- Little support for complex callback structures.
- Difficulty in fine adjustments for the layout such as label and text field alignments.
- Unclear support for intricate separation of interface from application code in complex synchronous and asynchronous dialogue sequences.
- No support for user defined widgets (such as the proposed drag-n-drop widget to be used in the recipients window).

At this point it was decided to abandon the use of XBUILD (and UIMS applications) for the development of the Iris UA. It is important to note that, at the time, XBUILD was one of the first generation UIMS applications for OSF/Motif and may have suffered from an immature development process and influenced by an early marketing strategy.

5.5 Design—Stage Three

The decision to discontinue use of UIMS development implied that the only suitable development process was direct C language implementation using the OSF/Motif widget library. This also proved serendipitous as a major redesign of the Iris UA was warranted when a greater rapport between X.400 MHS and X.500 Directory Services was identified.

5.5.1 Directory Searching

As a starting basis for development in this stage, it was decided to concentrate on the X.400 addressing and X.500 directory integration aspects of Iris [SAM-5.2/4, SAM-5.2/5]. Some initial work in this area is discussed in (Iannella, 1992b).

The first design (see Figure 70) involved the user entering a few key O/R address attributes; surname, given name, organisation, organisational unit, and country. These were selected since they represent the most common attributes in X.400 addresses. A directory search could then be performed using this information using either approximate or exact matching. Approximate matching uses algorithms to match with entries of similar spelling and pronunciation. If successful, the search results could be displayed in the lower text field of the window.

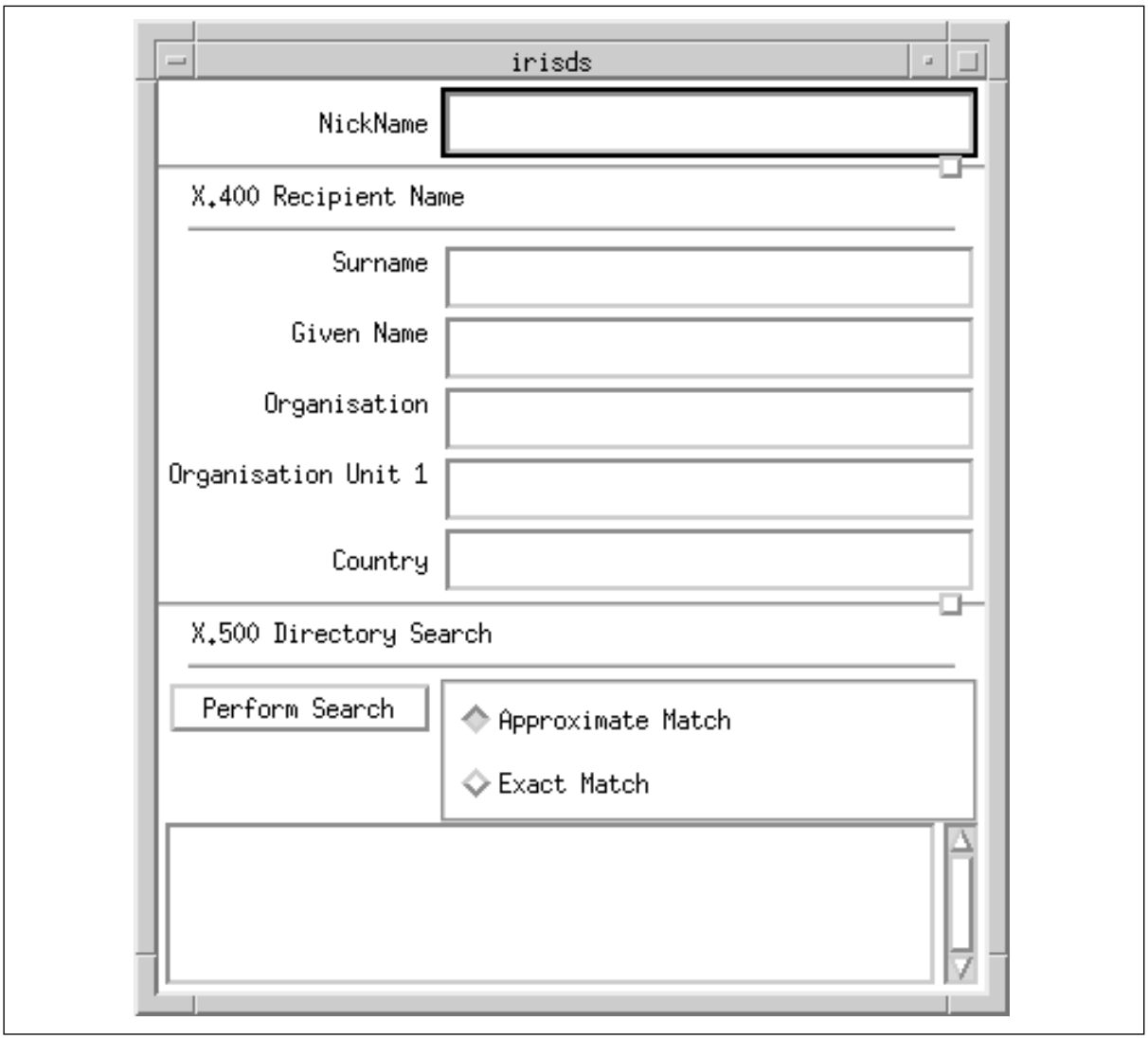


Figure 70: Initial Iris Directory Search Window.

Although the initial design proved successful, it was misleading in that the search for an X.400 address was based on O/R attributes when, in fact, they should be based on the recipient's X.500 entry attributes. It was also decided to execute all searches using the approximate matching algorithms as this produced greater success.

The next design divided the window into two sections to support both X.400 address entry and X.500 directory searching (see Figure 71). The left side of the window lists the full O/R address attributes which enabled the entry of a valid X.400 address. The shortform letter identifiers for each attribute are also displayed as these would feature prominently for an O/R address printed on a business card. The right hand side of the window consists of an X.500 directory browser.

The directory browser operates in four steps:

1 Select a country from the option menu.
2 Enter an organisation name and perform an approximate search of matching organisational names or return the full list of all organisations at the selected country level. The correct entry can then be selected from the matched list.
3 Enter a recipient's common name and perform an approximate search of matching personal entries at the selected organisational level. The correct entry can then be selected from the matched list.
4 The X.400 address of the selected personal entry can then be read and, if found, the O/R attributes on the left side of the window would be automatically entered.

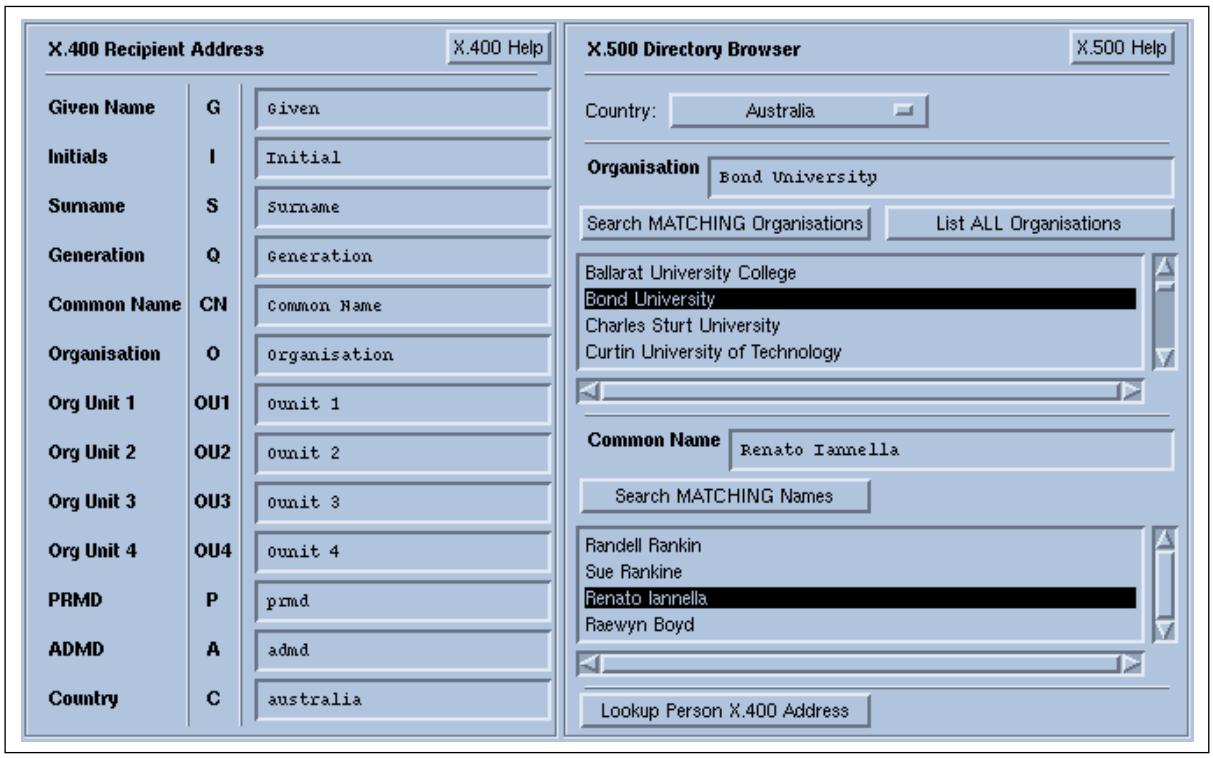


Figure 71: Iris Directory Browser

Although this directory browser interaction proved successful, it lacked simplicity and relied on a set series of steps to be performed in sequence. An easier interaction method was warranted but the directory browser was seen as a possible advanced option for the Iris UA.

It was clear that the 'intelligence' of the directory browser had to be integrated into the UA with minimal interaction from the user. This involves the development of a flexible algorithm for searching personal entries with maximum results. Similar algorithms exist (Wickham, 1992) but these assume that the distinguished name of the recipient is entered in freeform format. That is, the attributes could be in any order and are mainly targeted for text-entry interfaces.

Since LDAP was used for the X.500 directory searching, an appropriate algorithm was developed utilising its programming interface. The algorithm assumes that the user has entered the name (common name), organisation, and country of the person to search. These are the minimal requirements for any successful search. These values can be approximations or shortform versions of the full names. Also, since the country is selected from an option menu, the algorithm assumes that it is valid.

The Iris Directory Lookup Algorithm (IDLA) will search for the X.400 O/R address of a specified person using the following hierarchy:

1 Country
2 Locality (zero or more)
3 Organisation
4 Organisational Unit (zero or more)
5 Person

This structure implies that the simplest reference to a person should contain their name, organisation, and country. IDLA uses both exact and approximate searches for maximal success. Exact searches are performed first and if no matches are generated, then approximate matches are utilised. Considering the structure of the DIT, in most cases the three personal attributes (name, organisation, and country) are usually sufficient for a successful match. Other attributes that need considering are the Locality (eg states/provinces) and Organisational Units (eg departments).

IDLA utilises these two attributes to expand the search if errors or no matches are found. If no matching organisations are found then IDLA will assume that the organisation may be registered under a locality. IDLA will then search for all localities and present the list to the user for selection. If searching for a personal entry and an error occurs (time or size limit exceeded), IDLA will then search for all organisational units at that level and present the list to the user for selection. In both cases, IDLA will then continue the search with the new search base. Both the locality and organisational units attributes are searched recursively until no more entries exist. As an added benefit, IDLA will even search for localities under organisations or organisational units, a rare but possible scenario.

Table 20 summarises the possible IDLA search paths using both exact and approximate matching at each step. Appendix F shows a graphical representation of the Iris Directory Lookup Algorithm.

Table 20: IDLA Search Paths

Search Object	Search Paths
Organisation	Country (=Base A)
Organisation	Country, Locality (=Base B)

Table 20: IDLA Search Paths *(continued)*

Search Object	Search Paths
Person	Base A or B, Organisation
Person	Base A or B, Organisation, Org Units
Person	Base A or B, Organisation, Org Units, Locality
Person	Base A or B, Organisation, Locality
Person	Base A or B, Organisation, Locality, Org Units

5.5.2 User Agent Redesign

A major redesign of the Iris UA resulted in the discarding of the main window as this only provided an extra layer for accessing the functions of the UA [SAM-5.0/5]. The high-level separation of the Submission, Correspondence, and Administration functions could be avoided by providing direct and quicker access. The new main window now contains the message folders [SAM-5.0/9] and the message list. Returning to paper design, Figure 72 shows the new main window.

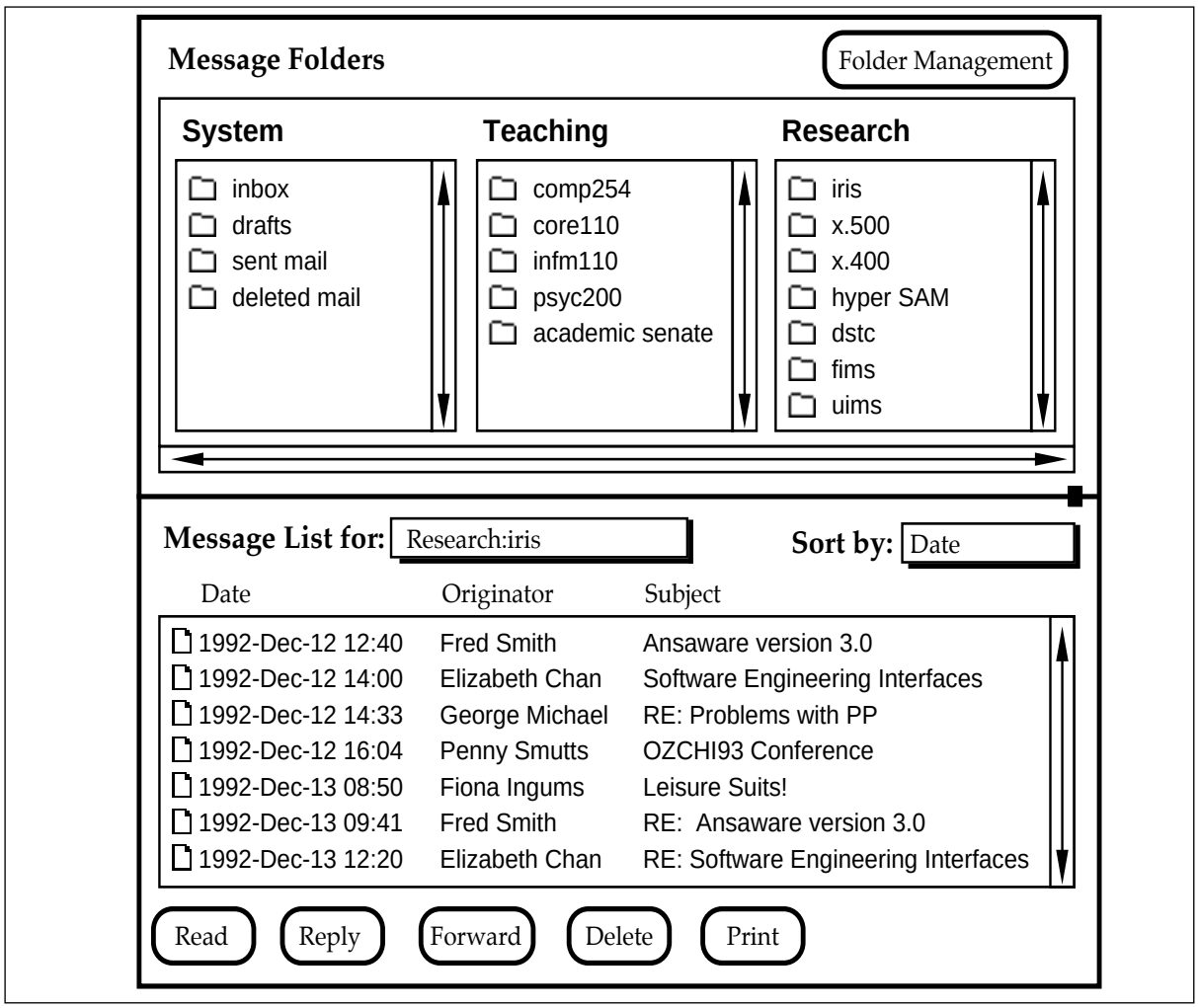


Figure 72: Stage 3—Iris Main Window Redesign

One of the issues of this design is the message folders list. Traditionally, message folder lists are single scrolling lists containing the names of the folders. These lists can grow enormously

over time. The system used on the main window categorises each group of message folders and incorporates a separate scrolling list for each [SAM-2.1/25]. The user is able to create new groups of message folders as well as new folders inside each group. The message folder group lists are also scrollable horizontally providing access to all the individual message folders. This new system for message folders allows the user to provide a categorisation structure for the folders instead of a large one-dimensional list. The structure is also appropriate for a two-level hierarchy. Hierarchies to greater levels would need alternative browsing structures to be designed. A number of default folders are provided such as 'inbox' for new messages [SAM-5.5/3].

The message list follows the standard of having a one-line summary of each message [SAM-5.5/9, SAM-5.5/12]. To change the message list to display the contents of a different message folder, the user simply double-clicks on the folder name in the message folder list. An option menu is also available to change to a different message folder. The option menu displays each folder group name and individual folder name. Another option menu allows the message list to be sorted by date, originator name, or subject [SAM-5.5/6, SAM-5.5/11]. Under the message list are a series of buttons that allow each message to be read, replied to, forwarded, deleted, or printed. To read a message, the user double-clicks on the message summary line or selects the message summary line and clicks on the Read button. A new window is then realised with the full message header and body parts displayed.

Another issue raised in this design is direct manipulation used for message archiving. Each message list item has an small document icon and each message folder is displayed with a small folder icon. The user can simply drag the message list document icon to a message folder icon to move a message. This is an efficient and suitable interaction technique for such a task. Unfortunately, the version of the OSF/Motif widget set used in the implementation did not support the drag-n-drop metaphor. A temporary alternative of a dialog window with a scrolling list of all message groups and folders was used.

The next major issue for redesign was the Send message window (Submission). The window must supply access to the following functions [SAM-5.1/9]:

- message subject,
- message body text,
- message options,
- message addressing with nicknames,
- message addressing with O/R addresses, and
- message addressing with X.500 directory searching.

The message subject and body text are catered for with the use of a single-line text field and a multi-line scrolling text field respectively. The message options were all grouped together on a separate dialog window. Figure 73 shows an example of one of the original designs. Most options utilised Toggle or Radio buttons for on/off or true/false values.

One of the message options features that this dialog fails to support is the concept of per-recipient options versus per-message options. The Importance and Sensitivity options are examples of per-message options. Their state applies to the entire message and for all recipients of that message. Notifications are classified as per-recipient options. The user must be

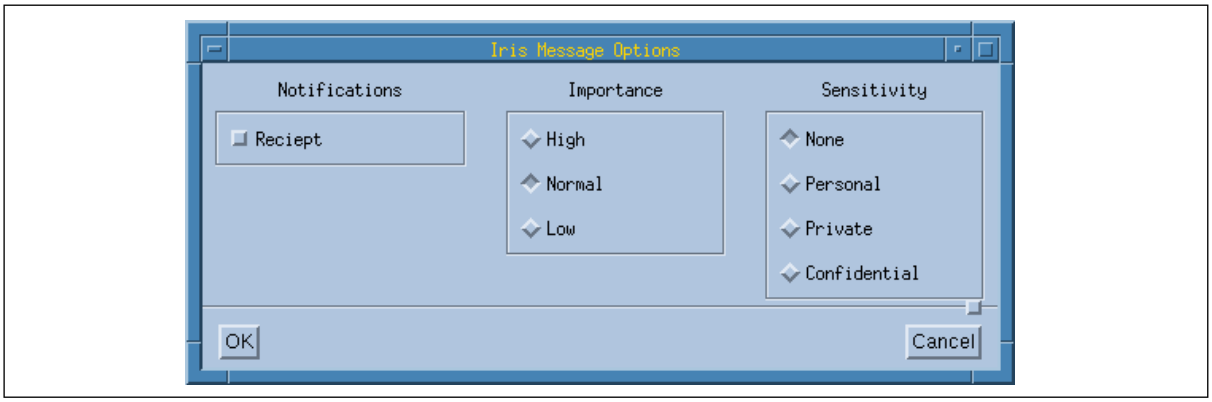


Figure 73: Stage 3—Iris Message Options (early design)

able to specify different notification options for each individual recipient of the message [SAM-5.3/11].

Probably the most difficult area of the Send message window is the design of the message addressing interaction. The user must be able to specify recipients of the message using predefined nicknames, full O/R addresses, or utilise an X.500 directory search. Also, each recipient must be categorised as Primary, Carbon Copy, or Blind Copy recipients. One of the early paper designs utilised a list of nicknames with three scrolling fields as the recipient categories and allowed direct manipulation to drag nicknames across to each field. Again, this could not be implemented due to the lack of support for drag-n-drop in the version of OSF/Motif used.

To specify a recipient using an O/R address, a new dialog window is employed that displays all attributes of the mnemonic O/R address and invites the user to fill in the text fields with the appropriate values [SAM-5.1/7]. The only attribute which is not specified with a text field is the country. In this case an option menu is used to select from the finite set of possibilities. The user also specifies the nickname to be associated with this O/R address.

To specify a recipient using an X.500 directory search, a new dialog window is displayed and the user enters the name and organisation of the recipient to search. Again, the country is selected from an option menu. Once the search begins, the IDLA is utilised to maximise the success of finding the recipient. If a match is found, the O/R address attribute of the entry is read from the directory and automatically stored. The user is then asked to specify a nickname for this new recipient.

The design and implementation process using the C and C++ language with the OSF/Motif widget library proved to be suitable for this project. The higher degree of control over the layout of the widgets and the interactive dialogues was evident. The extra confidence of the management of the callback routines (particularly for complex dialog interactions) was assuring and gratifying.

5.5.3 Final Screen Designs

The design and implementation process to develop the prototype Iris UA produced innovative and challenging screens. The final screen designs, prior to the usability evaluation process, are now presented and discussed. The overall screen map for the final window layout is show in Figure 74.

The main Iris window (see Figure 75) shows the message folders and the message list. One change made was the removal of the option menu for the list of message groups and folders.

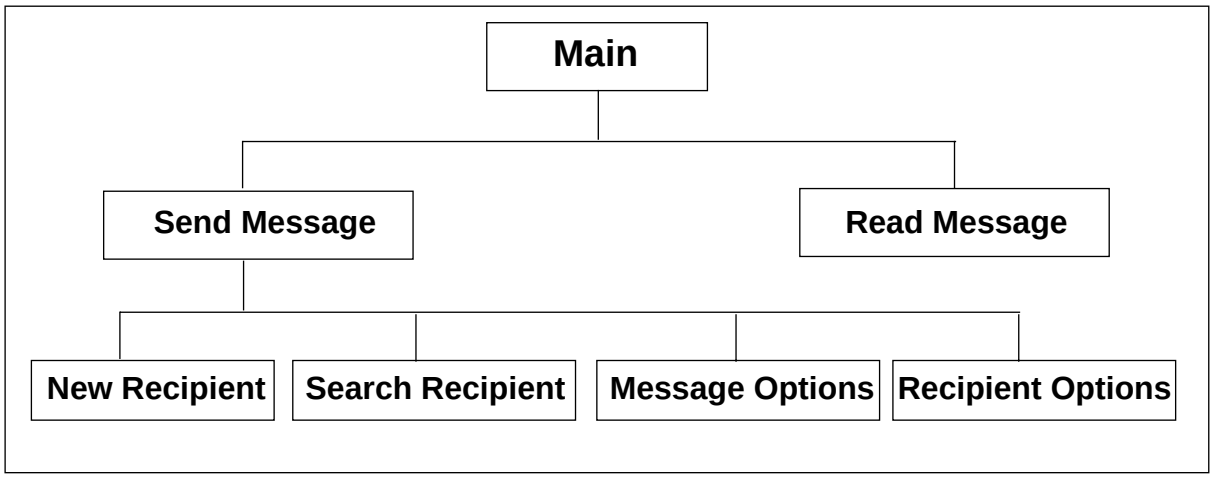


Figure 74: Stage 3—Iris Screen Map

The option menu was becoming too large to display on the screen and inconsistent with the original message folders list [SAM-3.1.3/20]. In its place is a text label that is updated to reflect the current message folder being displayed. To compose a new message for sending, the Send Message button at the top of the window is utilised. Two menu items, File and Help, are also available. The file menu simply has a Quit option, and the help menu contains a sub-menu of all the appropriate areas for the Iris UA. (Note: the help menu layout seemed appropriate for the prototype, the final version would have a help button on each main window.)

The control buttons on the bottom of the main window are invoked to read, reply to, forward, move, delete or print a selected message. The Read button displays the message content, and the Reply and Forward buttons set up a new message to edit and send. When replying to a message, the new message is automatically addressed to the originator of the message [SAM-5.2/15] and the Subject text field is preceded with 'RE:'. When forwarding a message, the new message's body is automatically filled with the original body text [SAM-5.2/19] and enclosed between a '—Forwarded Message—' heading and footer. The Subject text field is preceded with 'FWD:'. In the case of deleting [SAM-5.5/18] or printing [SAM-5.3/13] a message, a confirmation dialog is presented (see Figure 76). The default button options for these two dialogs is OK for printing and Cancel for deleting. Ideally, these two dialogs should also display the originator/subject of the message to be deleted or printed. Unfortunately, the OSF/Motif message dialog widget can only display one line of message text. This would have made the dialogs unacceptably wide. A custom dialog is warranted for the final version.

The Move button displays a dialog window with two scrolling lists (see Figure 77). The left list contains all the folder group names. When a folder group name is selected, the subordinate folder names are displayed on the right list. The user can then select the appropriate name from the folder list. Selecting the OK button would then move the selected message from the current folder to the new destination.

The read message window (see Figure 78) displays the message heading fields and the text body in a large scrolling field [SAM-5.5/14]. The control buttons on the bottom of the window are used in a similar fashion to those on the main window. The Close button removes the read message window.

The send message window presents the user with a range of options in composing a new message for submission (see Figure 79). The message recipients section of the screen contains a list

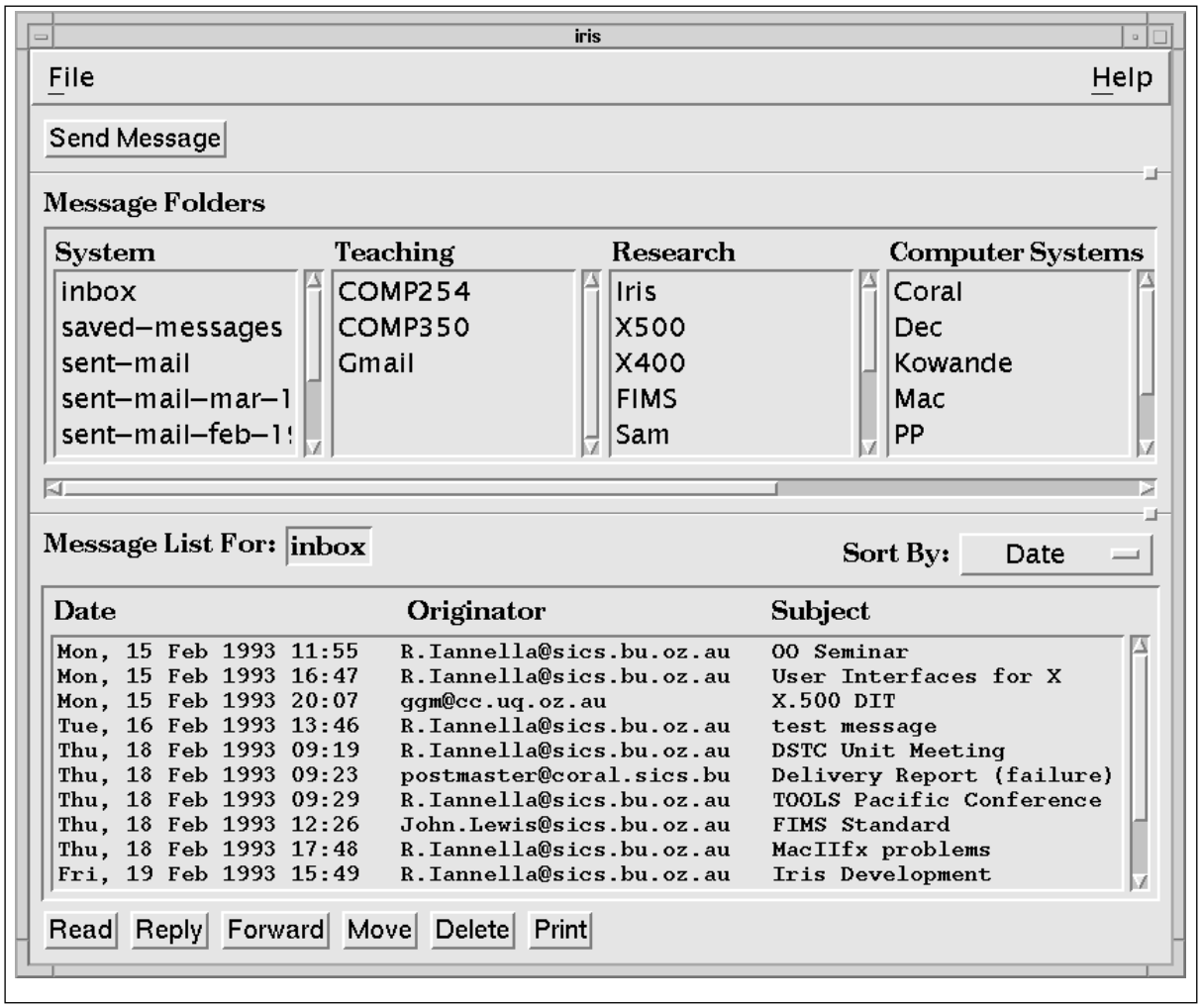


Figure 75: Stage 3—Iris Main Window

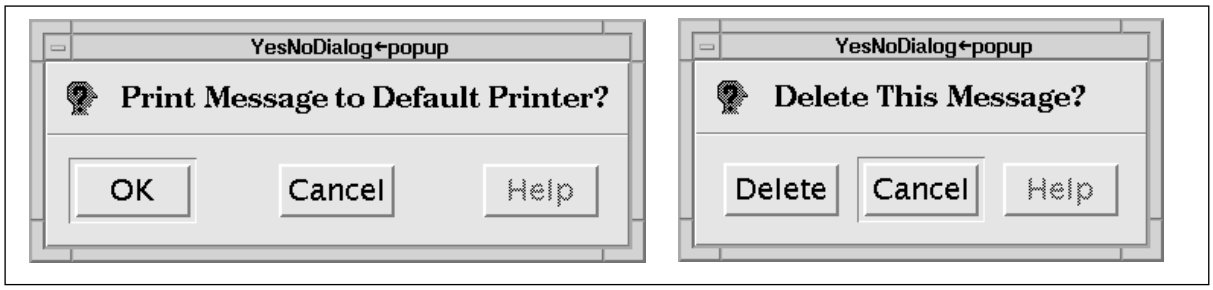


Figure 76: Stage 3—Iris Print and Delete Message Dialogs

of the predefined nicknames and two Add and Remove buttons. The Add and Remove buttons include the use of chevrons in the button name to indicate the direction of the move. The user can select a nickname from the list, select the Add button, and the nickname will move over to the message recipients list. Alternatively, the user may double-click on the nickname to achieve the same result. In both cases the nickname is removed from the nickname list, thus ensuring that a recipient occurs only once in a message [SAM-5.2/17].

The user also can specify the category of the recipient by selecting one of the Primary (the default), Copy, or Blind Copy toggle buttons first. The recipient category is also displayed with the nickname in the message recipients list. Nicknames in the message recipients list can also be removed by selecting the nickname and clicking on the Remove button (the nickname will

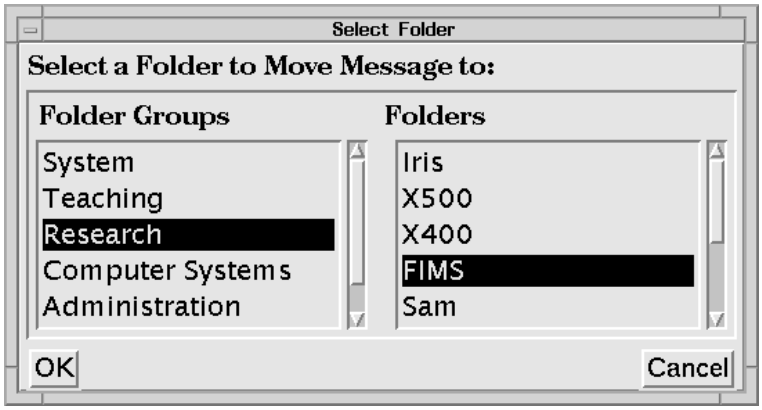


Figure 77: Stage 3—Iris Move Message Dialog

Figure 78: Stage 3—Iris Read Message Window

move back to the nickname list). Alternatively, the user may double-click on the nickname to achieve the same result.

Per-recipient options can be set by clicking on the Recipient Options button (only if a nickname in the message recipient list is selected first). Message options can be set with the Message Options button. New nicknames can be created with an O/R address (New Recipient Address button) or X.500 directory search (Search Recipient Address button). In each case, a new dialog is displayed.

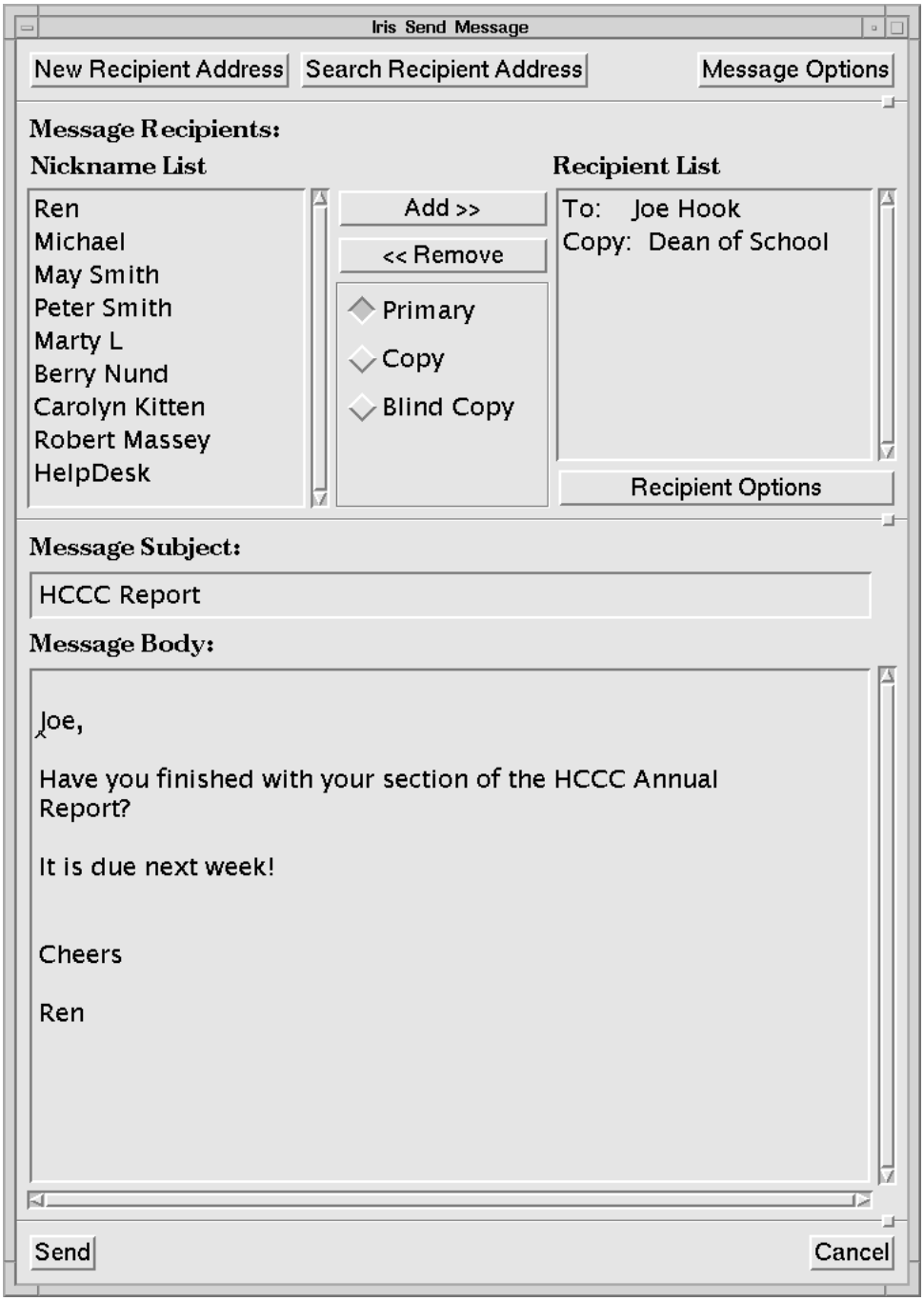


Figure 79: Stage 3—Iris Send Message Window

The bottom half of the send message window includes text fields to enter the message subject and body part. The subject utilises a one-line text entry field, and the body, a multi-line scrolling text field. Two buttons, Send and Cancel [SAM-5.3/12], allow the message to be submitted to the MTA or cancelled.

The message options window includes a number of facilities to manipulate the various service elements on an IPM (see Figure 80). The Importance and Sensitivity of the message can be indicated via a series of radio buttons. Appropriate default values are displayed. The Authorised By, Reply Recipients, Obsoleted Messages, and Related Messages options are all text

fields. The user may enter nicknames for the first two and IPM identifiers for the latter two into the text fields.

Figure 80: Stage 3—Iris Message Options Dialog

The message options also includes three toggle buttons to indicate the:

1 deferred delivery time,
2 message expiry time, and
3 reply to message by time.

Selecting any one of the toggle buttons will popup a dialog box for entry of the date and time of the option (see Figure 81). This dialog consists of a number of option menus to enable the quick selection of the date and time values. The full range of days, months, and hours can be selected. The years are limited to a small number of future years and the minutes option menu displays five minute increments. The default date is set to the end of the current year (or a pre-defined offset) and the default time is 12:00 AM [SAM-1.8/1].

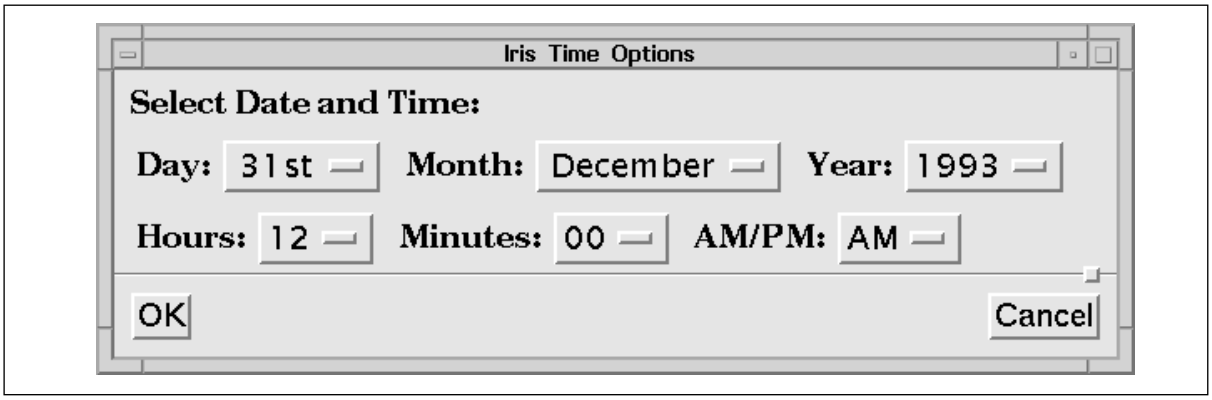


Figure 81: Stage 3—Iris Date & Time Options Dialog

The per-recipient options dialog consists of a number of toggle buttons to set the notification level for the selected recipient (see Figure 82). The dialog displays the message recipient's nickname along with four toggle buttons to set:

1 non-receipt notification [SAM-5.4/7]
2 receipt notification [SAM-5.4/2]
3 return message body (if message cannot be delivered)
4 reply requested (from originator)

By default, only the non-receipt notification toggle button is set to be on. The second and third options are indented under non-receipt notification indicating that they are dependent on this option being true. If non-receipt notification is turned off, then the two subordinate options are made insensitive and nonoperational.

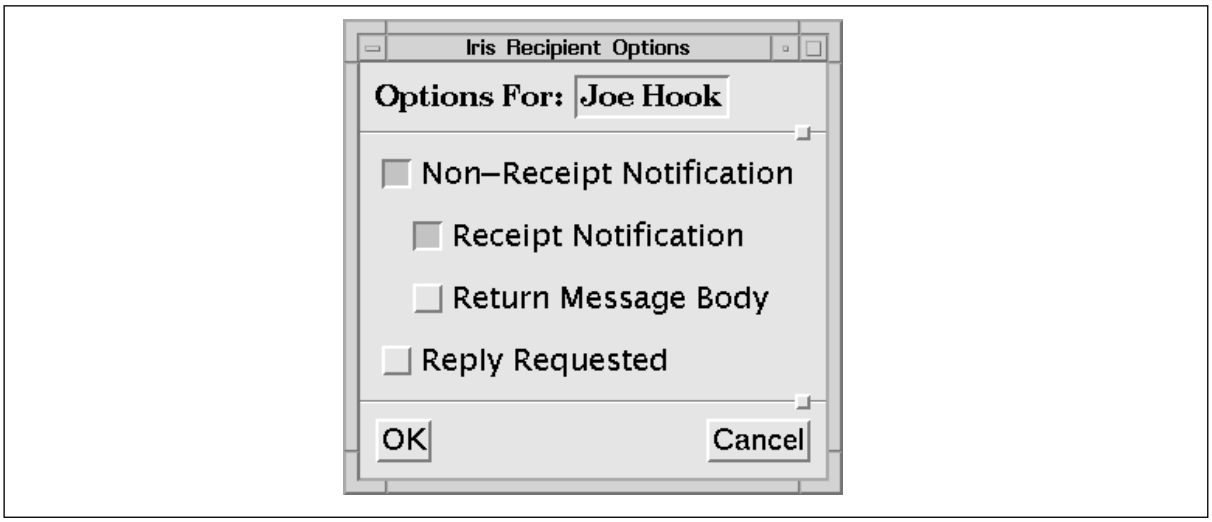


Figure 82: Stage 3—Iris Recipient Options Dialog

The Enter Recipient Address window displays the full list of attributes for the mnemonic O/R address (see Figure 83). Each text field is preceded with an associated label [SAM-1.4/5] and entry cue [SAM-1.4/18]. The label includes the common abbreviation for the attribute to aid in transcribing O/R addresses from business cards and other sources [SAM-1.4/18, SAM-1.4/19]. The country is specified from an option menu.

At the top of the window, a user supplied nickname is entered in the text field to be associated with the O/R address recipient. This user can specify the nickname to be any text string that

can be subsequently used to identify the recipient. The new nickname is automatically added to the message recipients list.

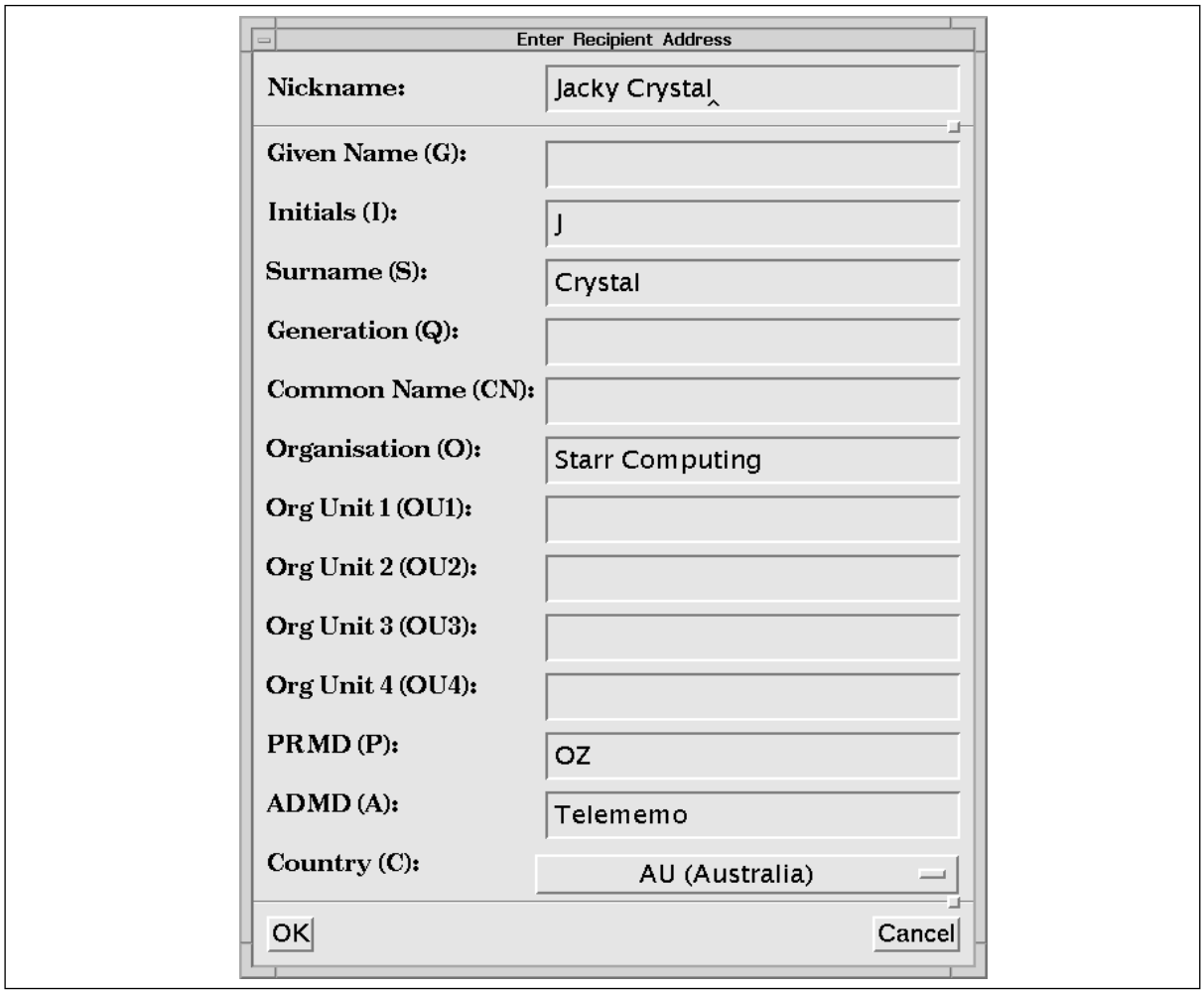


Figure 83: Stage 3—Iris O/R Address Form Dialog

The Search Recipient Address window asks the user to enter the name and organisation of the potential recipient (see Figure 84). These values may be approximate matches as the IDLA will maximise the search success. The country is selected via an option menu. Since the searching of the X.500 directory can be a time consuming task, appropriate feedback is used to inform the user of the progress of the search [SAM-4.2/1].

Figure 84: Stage 3—Iris Directory Search Dialog

Given the example information depicted in Figure 84, the IDLA process will firstly establish connection to the LDAP server and then commence the first search for the organisation. The appropriate informative Working dialogs that are displayed are shown in Figure 85.

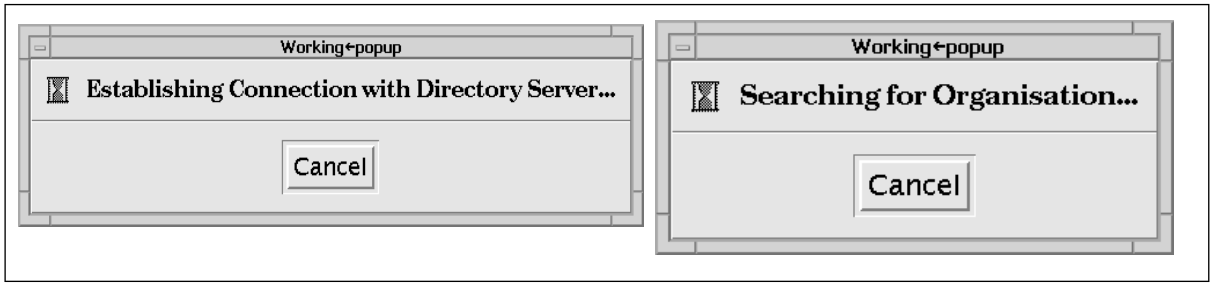


Figure 85: IDLA Connection and Search Feedback Dialogs

If multiple matches are made on the organisation, then the user is presented with a new dialog listing all the approximate matches (see Figure 86). The user may then select the correct match to continue the search or cancel the entire search.

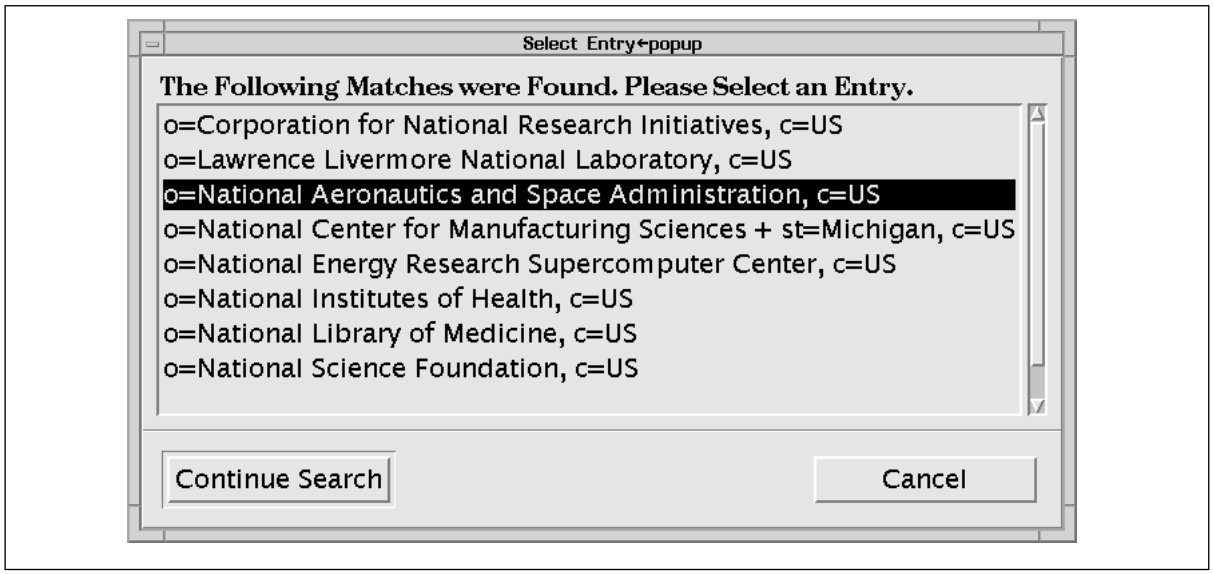


Figure 86: IDLA Multiple Organisation Selection Dialog

If IDLA finds a single matching organisation or the user has selected one, the search continues looking for the personal name (see Figure 87).

Figure 87: IDLA Name Feedback Dialog

If, at any level of the IDLA search, no match is found, then the user is informed and the search is cancelled (see Figure 88).

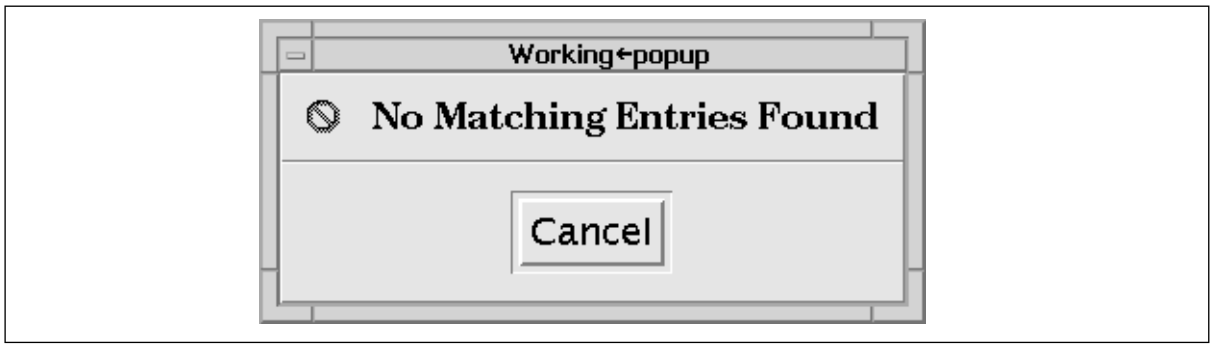


Figure 88: IDLA No Matching Entries Found Dialog

If multiple matches are made for the name, then the user is again presented with a dialog listing the matches and asked to select one. The search can then be continued or cancelled (see Figure 89).

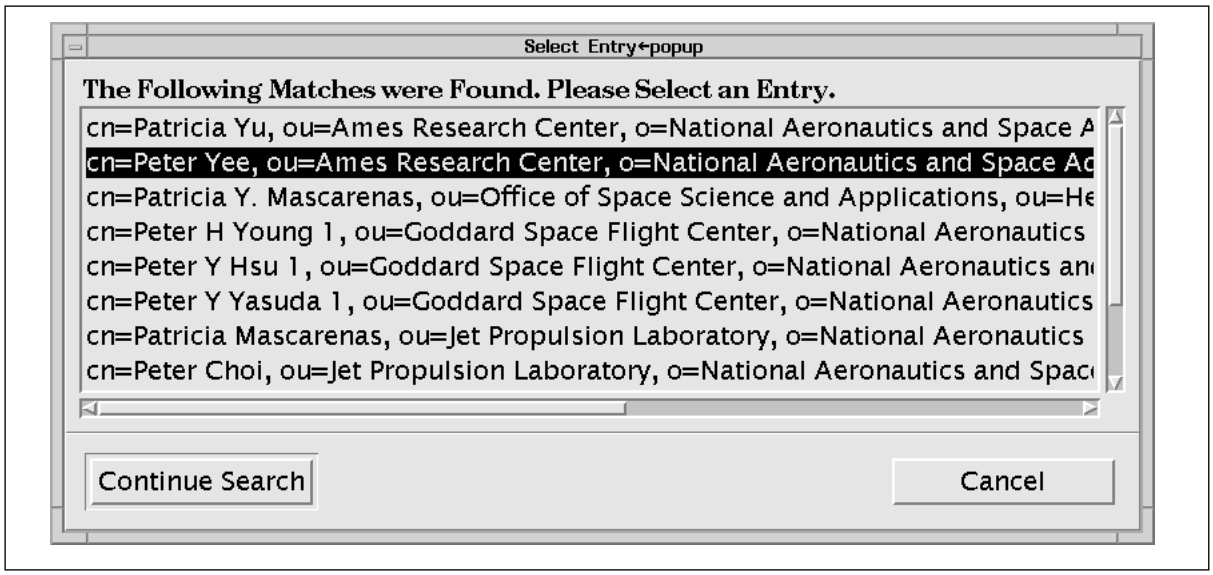


Figure 89: IDLA Multiple Name Selection Dialog

If IDLA finds a single matching name, or the user has selected one, it then attempts to read the entries O/R address attribute. If successful, the user is asked to enter a nickname for this recipient and the nickname is automatically added to the message recipient list (see Figure 90).

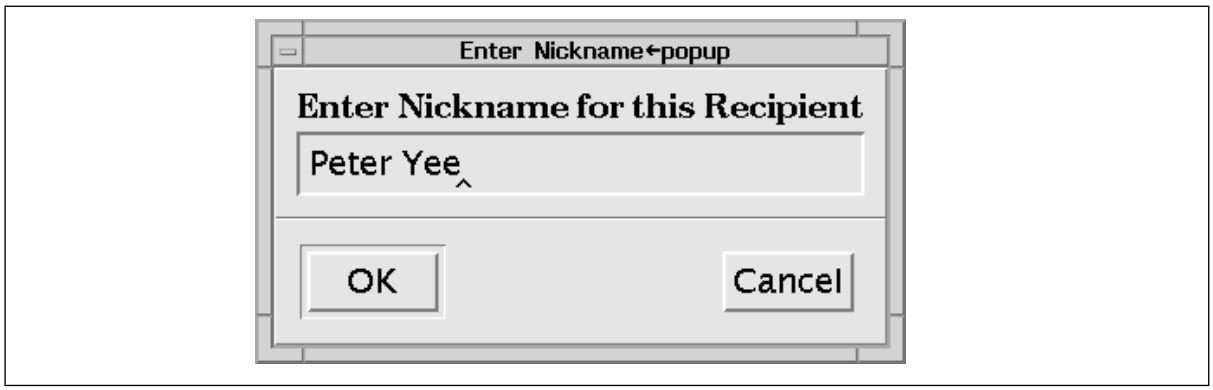


Figure 90: IDLA Nickname Dialog

If the entry has no O/R attribute then the user is informed (see Figure 91).

Figure 91: IDLA No O/R Address Found Dialog

5.6 Summary

During the design and implementation, appropriate testing of each new dialog and function was undertaken. After the completion of the final screen designs, extensive examination of the complete Iris UA prototype[17] was performed. All possible scenarios were tested with the interface to ensure the UA would fulfil all the intended functionality requirements. The X.500 directory interaction and the IDLA was comprehensively analysed and results verified.

The design process required extensive knowledge of the X.400 IPM service elements, and hence, an understanding of the X.420 protocol. Interpreting the functional protocol into an appropriate user interface was a non-trivial task. The X.500 directory searching routines, provided by the LDAP routines, enabled the addressing of messages with an intelligent searching engine.

The implementation of the Iris UA using the OSF/Motif C library routines proved a very successful and flexible environment. The control of the user interface was more evident than compared to the less successful use of the XBUILD UIMS. The lack of support for drag-n-drop in the version OSF/Motif used required the redesign and implementation of some interaction techniques. Nevertheless, the final graphical user interface favourably utilised the OSF/Motif interface widgets.

[17] Approximately 5,000 lines of C/C++ code.

Chapter 6

Evaluation and Usability

6.1 Introduction

The evaluation of the Iris interface involved the selection of appropriate and effective evaluation methodologies. Each method has advantages and disadvantages, both in terms of time and costs involved to achieve varying levels of results. See section '2.8 Evaluation and Usability' on page 42 for an outline of the major usability evaluation techniques and their characteristics.

For the evaluation of the Iris UA, three methods were selected on the basis of cost-effectiveness and identification of user interface problems. This selection process was controlled by the resources available for the evaluation and the level of expertise required to administer each method. The evaluation methods selected included:

- scenario-based usability trials
- thinking-aloud method
- user questionnaires

Studies have shown that the use of early paper-based evaluations with scenarios and computer-based prototypes is an effective method for detecting usability problems (Nielsen, 1990b).

Usability evaluation offers substantial feedback for interface evaluation by providing direct access to system use by real users. A scenario-based usability evaluation is described which captures the functional usage of the interface. The scenario describes a number of common tasks in the daily use of electronic mail systems. The identified tasks cover a range of user activities that should be supported by X.400 messaging systems. Each task also reflects targeted functionalities of the Iris UA.

The usability tests are video taped which allows a complete and thorough analysis of each user's interaction with the Iris UA. The thinking-aloud method is employed by the trialists to provide a comparison of the user's mental model with the interface's model. A series of questionnaires is used to back up the findings of the analysis and provide a statistical analysis

across the usability trials. The method used to analyse the video tapes is discussed as it provided an integral part of the evaluation method.

A summary of the usability evaluation results is used to identify aspects of the Iris UA interface that may require redesign. The Iris UA prototype is then modified to reflect the new interface design changes. The usability evaluation process is repeated on the redesigned interface in an iterative manner. The results from the second evaluation are then discussed and matched with the original interface problems.

An analysis of the questionnaires is performed, utilising statistical testing methods, to establish any significant differences between the usability trials. A high level summary of the questionnaire results is also discussed to establish overall achievements and success of the evaluation process. Finally, a discussion of the comparative times each trialist took to complete each task is presented to contrast each evaluation group. Both the task times and the questionnaire analysis support the usability evaluation methodology.

6.2 Usability Evaluation

Prior to the design and evaluation of the Iris prototype, a set of clear usability goals was developed (see '2.8.1 Usability' on page 42). These goals also helped focus the design of the user interface. Nielsen (1992) defined a series of goals that were clearly useful for this evaluation. These usability goals included the following:

- learnability,
- efficiency of use (once the system has been learned),
- ability for infrequent users to return to the system without large re-learning,
- reduction of the frequency and seriousness of user errors, and
- subjective user satisfaction.

The selected usability evaluation methods involve a process of observing users interacting with software whilst being video taped. The video allows the evaluation to be reviewed any number of times at a later stage. If the review of the video highlights problems with the interface, then a subsequent usability evaluation can be performed after a redesign of the original interface. This process can then continue in an iterative manner. Video is also useful for developers to see their software in use and provides a permanent record of the events. Figure 92 shows a typical layout for the usability trial.

The taping of the trialist provides a convenient mechanism for the review of the user interactions with the software. The video can be scrutinised any number of times. As an added benefit, a small mirror was added to the top-left of the computer monitor to provide feedback of the eye movements of the trialist (Perlman, 1992b). This provides extra information as to where the trialist is looking on the screen. The facial expressions are also an important indicator as to the user's understanding of the interface. The trialists also used the thinking-aloud method to further provide feedback on what they were thinking during each session. A clip-on microphone was used to capture the audio input.

Since each session was being video taped, guarantees had to given to the trialists to protect their privacy. Each trialist was asked to sign a consent form to this effect. This procedure also followed Bond University guidelines for experimentation involving human subjects. The importance of this process cannot be overstated as it protects all parties concerned and outlines

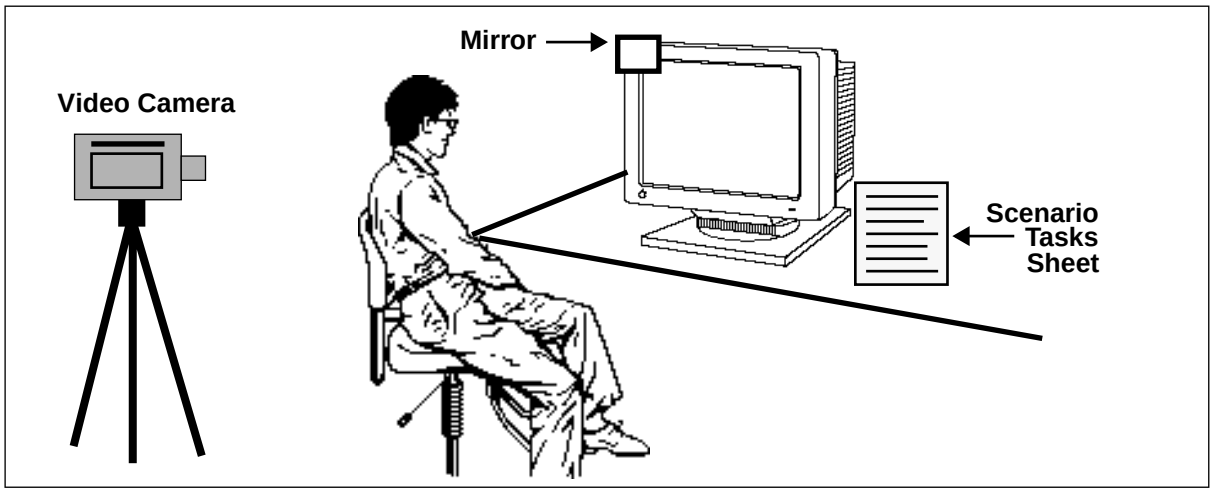


Figure 92: Usability Evaluation Setup

the objectives of the evaluation. Assurances are made that the public release of data generated from the evaluations will not identify any individual trialist. The consent form is listed in Appendix G and is an adaptation from Perlman (1992b).

For the evaluation of Iris, the trialists were asked to perform a number of typical tasks that are common for electronic mail users. These tasks form a benchmark in the final evaluation of the interface. The trialists were also asked to complete a questionnaire about the functionality and the graphical interface of Iris after each session.

6.2.1 Scenario Description

The scenario developed for the usability evaluation reflect five common tasks that are typical of daily use of electronic mail applications. Nielsen (1990b) defines the main characteristics of a scenario task as encapsulating the description of:

- a single user,
- using a specific set of computer facilities,
- the achievement of a specific outcome, and
- specific circumstances.

Each task is representative of realistic activities of an electronic mail user and cover the major functions of the Iris UA. Table 21 lists the five tasks used in the evaluation.

Table 21: Usability Scenario Tasks

No	Task Description
1	1—Read the mail message from John Lewis about the FIMS standard. 2—Reply to the message and ask him to send you a copy of the FIMS standard. 3—Move the message to the folder about FIMS standards
2	1—Find and read the mail message that you received last month from Lisa Naven—it was about your lecture times for the COMP254 subject next semester. 2—Print the message and delete it.

Table 21: Usability Scenario Tasks *(continued)*

No	Task Description
3	You and your work colleague Joe Hook are working on the HCCC Annual Report. The report is due soon and you wish to send Joe a message asking if he has finished with his part of the report. You also require the following to be specified before sending the message: 1—Since this is urgent and highly important, indicate that you wish to receive automatic notification of when he reads this message. 2—Also indicate that you wish to receive a reply from him by the 1st July 1993. 3—Make sure a copy of the message gets sent to the dean of the school.
4	You have been given a business card of an interstate colleague: Starr Computing *1 Lexington Dr* *Prospect* *Sth Aust, 5082* *Ph 344 4455* **Jacky Crystal** Software Engineer X.400: I=J; S=Crystal; O=Starr Computing P=OZ; A=Telememo; C=AU 1—Send her a message and ask if you could visit the organisation next month with view to some collaborative research.
5	You have read an interesting research paper on a user interface design tool for astronauts by Peter Yee from the Ames Research Center at the National Aeronautics and Space Administration (NASA) in the USA. 1—Send him a message asking if the user interface design tool is available for evaluation.

All the names, organisations, and activities depicted in the tasks are fictitious and were developed to mimic real-life situations, except for task five. This mentions a real person, Peter Yee from NASA, who agreed to allow his X.500 directory entry to be used in the experiment. This enabled the directory addressing interaction to be as realistic as possible since it included the time taken for the directory lookup and the feedback sequences.

Each task was designed to utilise and test certain features of the Iris UA in typical real-world situations. The tasks offer challenges to the trialists in transferring each messaging activity to the Iris UA interface. The first two tasks reflect common read, send, and message management activities. Task one asks the trialist to read a message (currently listed in the inbox folder) and reply to the message. The original message is then moved from the inbox to another folder. Task two involves searching for an existing message in a different message folder. Once found, the message is then read, printed, and deleted.

Task three is a more complex example involving addressing a message to a number of users (in different recipient categories). The recipients in this task are all predefined nicknames. The

task also involves setting various IPM options. The options involved are the most common service elements that an IPM user would utilise.

The final two tasks involve creating new addresses (nicknames) for messages in two different styles. In task four, the trialist was given a business card of a colleague and asked to send her a message. The business card lists the recipient's X.400 O/R address in the recommended format. The task involves translating the O/R address from the business card to the form interface provided by Iris. Task five requires the searching of an X.400 address using X.500 Directory Services. The trialist is asked to send a message to a specific recipient given the name, organisation, and country of that person. The task requires the user to formulate a directory lookup and interact with the IDLA sequence of dialog screens.

6.2.2 Questionnaires

A number of questionnaires were formulated to elicit quantitative measures from the usability trials and provide an indication of the background of the trialists. Questionnaires are an easy and quick method to provide extra feedback for the user interface evaluation. The first questionnaire asks the trialist questions on the number of years experience they have had with computers and various electronic mail systems (see Table 22). This small questionnaire establishes the history and ensures a common background of the trialists.

Table 22: Usability Trialist Experience Questionnaire

Select the number of years experience with the following (Place a ✔ in the appropriate column)	None	Less than 1 Year	Less than 2 Years	Less than 3 Years	Greater than 3 years
Experience with computers					
Experience with text-based interface electronic mail systems					
Experience with graphical interface electronic mail systems					
Experience with X.400 electronic mail systems					

The second questionnaire focuses on the functionality of the Iris UA. The trialist is asked to indicate their level of agreement with a number of statements about the clarity of each messaging function that was involved in the usability evaluation. Five levels of agreement are possible, from 'strongly disagree' to 'strongly agree'. Finally, the trialists are asked about the overall provision of services in the Iris UA. Each question is related directly to the scenario and the specific tasks that each trialist was asked to perform. Table 23 shows the questionnaire for the functionality of the Iris UA.

Table 23: Usability Functionality Questionnaire

Rate your agreement with the following statements (Place a ✔ in the appropriate column)	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
It was clear to…					
read a mail message					

Table 23: Usability Functionality Questionnaire *(continued)*

Rate your agreement with the following statements (Place a ✔ in the appropriate column)	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
reply to a mail message					
send a mail message					
move a mail message to a folder					
move to different mail folders					
delete mail messages					
print mail messages					
It was clear on setting…					
message options					
recipient options					
It was clear on addressing mail messages…					
using the nicknames					
given an X.400 address					
using directory searching					
Overall, the system provided…					
adequate functions for electronic mail					
easy access to all options for electronic mail					
support for appropriate electronic mail tasks					

The third questionnaire focuses on the graphical user interface of the Iris UA. The trialists are asked to indicate their agreement with a number of statements about aspects of the Iris UA interface. Five levels of agreement are possible, from 'never' to 'always'. The questions were taken from a small subset given in (Ravden & Johnson, 1989). Ravden and Johnson provide a detailed checklist of numerous questions grouped into nine sections. For the interface questionnaire, only two questions from seven of the sections were selected. These questions provided a small and concise set which was specifically targeted at the interface under evaluation.

Finally, the trialists are asked about the overall graphical user interface of the Iris UA. Table 24 shows the questionnaire for the interface of the Iris UA.

Table 24: Usability Interface Questionnaire

Section	Rate your agreement with the following statements (Place a ✔ in the appropriate column)	Never	Some of the Time	Neutral	Most of the Time	Always
Visual Clarity	Does the information appear to be organised logically on the screen					
	It is easy to find the required information on the screen					
Consistency	Is the information (eg menus, buttons, lists) displayed in a consistent location and layout					
	Is the way the system respond to a user actions consistent at all times					
Expectations	Is the format of displayed information compatible with the users perception					
	Does the sequence of activities required to complete a task follow what the users would expect					
Feedback	Are messages and instructions relevant and clear to the user					
	Are status messages informative and accurate					
Explicitness	Where the user is presented with a list of options, is it clear as to what each option means					
	Is it clear what different parts of the system do					
Flexibility	Do users have control over the order of information entry					
	Is the system flexible in allowing the user to choose options					
Error Prevention	Does the system allow the user to easily correct data					
	Does the system protect against errors in user actions					
	Overall, the interface was pleasing and easy to use					

The two latter questionnaires focus on:

1 functionality, and the
2 graphical user interface.

It was important to present these two aspects as separate questionnaires to best elicit the trialists responses. If the questionnaires were combined, it would be difficult to ascertain if the responses were highlighting problems in the interface or lack of functionality.

6.2.3 Usability Evaluation Method

The trialists for the evaluation were undergraduate computing students who all had experience of using graphical and non-graphical electronic mail systems. None had previous experience with X.400 MHS. See Appendix H for a summary of the trialist experience questionnaire. The trialists were paid $15 per hour for their participation in the evaluation which took no longer than one hour each. Each trial consisted of six students. A sample size of six is double the minimum recommended number by Perlman (1992b) for observational studies and would also allow for quantitative differences to be analysed.

The following steps were taken for each trialist:

1 Explain the concept of usability evaluations and the purpose of the research.
2 Introduce the Iris UA prototype.
3 Reconfirm that the trial will be video taped.
4 Explain that the trial is to test software not users.
5 Explain consent form and emphasise the confidentiality clause.
6 Issue the first experience questionnaire.
7 Explain the five tasks.
8 Explain (with an example) the thinking-aloud method.
9 Run video and check microphone.
10 During the trial, prompt the use of the thinking-aloud method if the trialist seems 'lost'.
11 When finished, stop video.
12 Issue the functionality and interface questionnaires.
13 Ask the trialist not to divulge the content of the evaluation to other people (particularly other trialists).
14 Thank and reimburse the trialist.

A dry run was first performed on a test trialist to gain experience and to get any feedback on the sequence of steps or any other aspect that may have been unclear.

After each trial, a comprehensive review of the video tape was undertaken to analyse each session. The video tape proved indispensable in the detection of user problems. These were revealed by noticing hesitations, errors, uncertainty, and redundant moves and selections made by the user. The eye movements and facial expressions were clear indicators as to when the user was experiencing these difficulties. Matching the current task that the user was attempting (from the user 'thinking-aloud') with such difficulties enabled detailed usability problems to be noted. Even though the video recordings enabled this analysis to be effectively performed, including slow motion and pause facilities, the vast amount of time required was considerable.

The sequence of steps performed was recorded on paper as the trialist performs each task. Also, the time taken to compete each task is noted. Table 25 shows an example of the level of de-

tail recorded for each trialist. The example shows a a trialist's attempt at task three and lists all the actions and possible problems.

Table 25: Usability Task Recordings Example

Sub-Task	Description
Send New Message	• Looked in HCCC folder • Looked at buttons on bottom of main window • Found Send Message button (at top of window)
Address Message	• Select Joe Hook in nickname list • Clicked on Add button
Message content	• Entered message subject • Entered message body
Message Options	• Clicked on Recipient Options button—no response • Clicked on Message Options button • Selected High Importance toggle button • Selected Reply to Message By toggle button—new dialog appeared • Selected correct date from option menus
Address Message (2)	• Selected Dean of School nickname • Clicked on Add button • Realised incorrect—clicked on recipient name and Remove button • Selected Copy radio button and added correctly
Message Options (2)	• Clicked on Recipient Options button—no response (again) • Click on Joe Hook in recipient list • Clicked on Recipient Options button • Selected Receipt-Notification and Reply Requested toggle buttons
Send message	• Clicked on Send button

After the completion of the video analysis for each task and trialist, a higher level summary was generated. This summary concentrated on the problems experienced by the trialists that were attributed to the user interface. These points were extrapolated from the video transcripts that identified the user interface problems in each task. Each point in the summary was numbered to aid in the cataloguing of the task difficulties. The new Iris interface could then be redesigned to address each of these user interface problems.

There are three levels of problem categories that can be found with usability evaluations (Perlman, 1992b):

1 High-level. Difficulties with task and not the user interface.
2 Medium-level. Problems accomplishing task because of the user interface.
3 Low-level. Easy to fix user interface problems.

An example of a high-level problem is misunderstanding the task description. This was evident in task three in which a number of trialists were unsure of the task requirements. Instead of sending a new message, they were inclined to search for an existing message from or about Joe Hook or the HCCC annual report. The majority of problems found in the usability evalu-

ation fall into the latter two categories and required minor to substantial changes in the user interface of Iris.

6.3 Usability Trial One Results

Upon completion of the first trial and the analysis of the videos, a summary of user interface problems was generated. The summary appears in Table 26 and is referenced by the task number and the identified problem number. The summary also includes a classification of the problem into the three levels.

Table 26: Usability Trial One—Interface Problems Summary

Task Number	Problem Number	Description of User Interface Problem	Problem Level
1	1	Could not find the message to read—looked in message folders first before looking at message list.	Medium
	2	Clicked on Read button without selecting message first.	Low
	3	Message list text is too small.	Low
	4	Move message dialog was confusing. User wasn't sure what to do with the two lists.	Medium
2	1	Difficulty in finding required message. User did not realise that the subject of the message was the message folder name.	High
3	1	Send Message button was not obvious.	Low
	2	Users tended to look for a message about Joe Hook or the HCCC Annual report in a message folder rather than send a new message.	High
	3	Recipient Options button provided no feedback when no recipient is selected.	Low
	4	Added 'Dean of School' as primary recipient instead of copy recipient.	Low
	5	Difficulty in finding Message options and confusion with the differentiation between Message and Recipient options.	Medium
4	1	Did not enter a nickname.	Low
	2	Filled in O/R address from information on the business card except the X.400 address listed.	Medium
	3	Left out some fields when filling in the O/R address.	Medium
	4	Filled in too much data into the O/R address.	Medium
	5	Unsure as to the meaning of the O/R address field acronyms.	Low
	6	Some users were looking for an Internet style address to enter.	High
	7	Some users tried 'Search Recipient Address' instead of 'New Recipient Address'.	Medium

Table 26: Usability Trial One—Interface Problems Summary *(continued)*

Task Number	Problem Number	Description of User Interface Problem	Problem Level
5	1	Some users tried 'New Recipient Address' instead of 'Search Recipient Address'.	Medium
	2	Did not enter a nickname even when presented with a dialog box.	Medium
	3	'Continue Search' button led to confusion after matched names are presented for selection.	Low
	4	Rewording of 'Select an Entry' dialog box.	Low
	5	Feedback as to the meaning of X.500 attribute acronyms.	Low

The summary enables the redesign of the Iris interface to be structured and focused on the particular problems it raises. The problems identified as high-level were unable to be solved with interface changes as they reflected misconceptions with the tasks. It was inadvisable to change the tasks at this point as it would than defeat the purpose of a comparison across two groups of trialists.

6.3.1 Iris UA Interface Redesign

From the identified interface problems listed in Table 26, a number of changes were made to most of the screens and dialogs in Iris (all except the Read message screen). The first interface screen to undergo redesign was the main window. The major change was to make the message list the central focus of the main window by moving it to the top half of the screen and moving the message folders to the bottom half. The font size of the message list was made larger which required the Date and Originator fields to be reduced in size slightly. These changes address interface problems 1–1 and 1–3.

Another change on the main window was the renaming of the 'Send Message' button to 'Compose New Message' to address problem 3–1 by offering a more descriptive title to the button's function. Figure 93 shows the new redesigned Iris main window.

Another small modification to the main window was the addition of a feedback dialog if the user selected one of the buttons under the message list with no message selected. In this case, the user is informed that a message needs to be selected first. This change addresses interface problem number 1–2 and is shown in Figure 94. Another solution to this problem is for the system to automatically select the first message by default. This has the disadvantage of not allowing the user to be in full control and will led to inconsistencies if a message is deleted.

The move message dialog proved difficult to use in the first design of Iris and a number of options were considered for the redesign. One design was to incorporate an option menu of message folder groups which then displayed the subordinate list of message folders. This, as with the original design, still requires two actions by the user to find the appropriate message folder which may have led to the dialog's poor usability result.

The final redesign was a single outline list that incorporates all the message folder groups and the message folder names. The message folder group names would be followed by a colon and the subordinate message folders would be listed below (with appropriate indentation). This

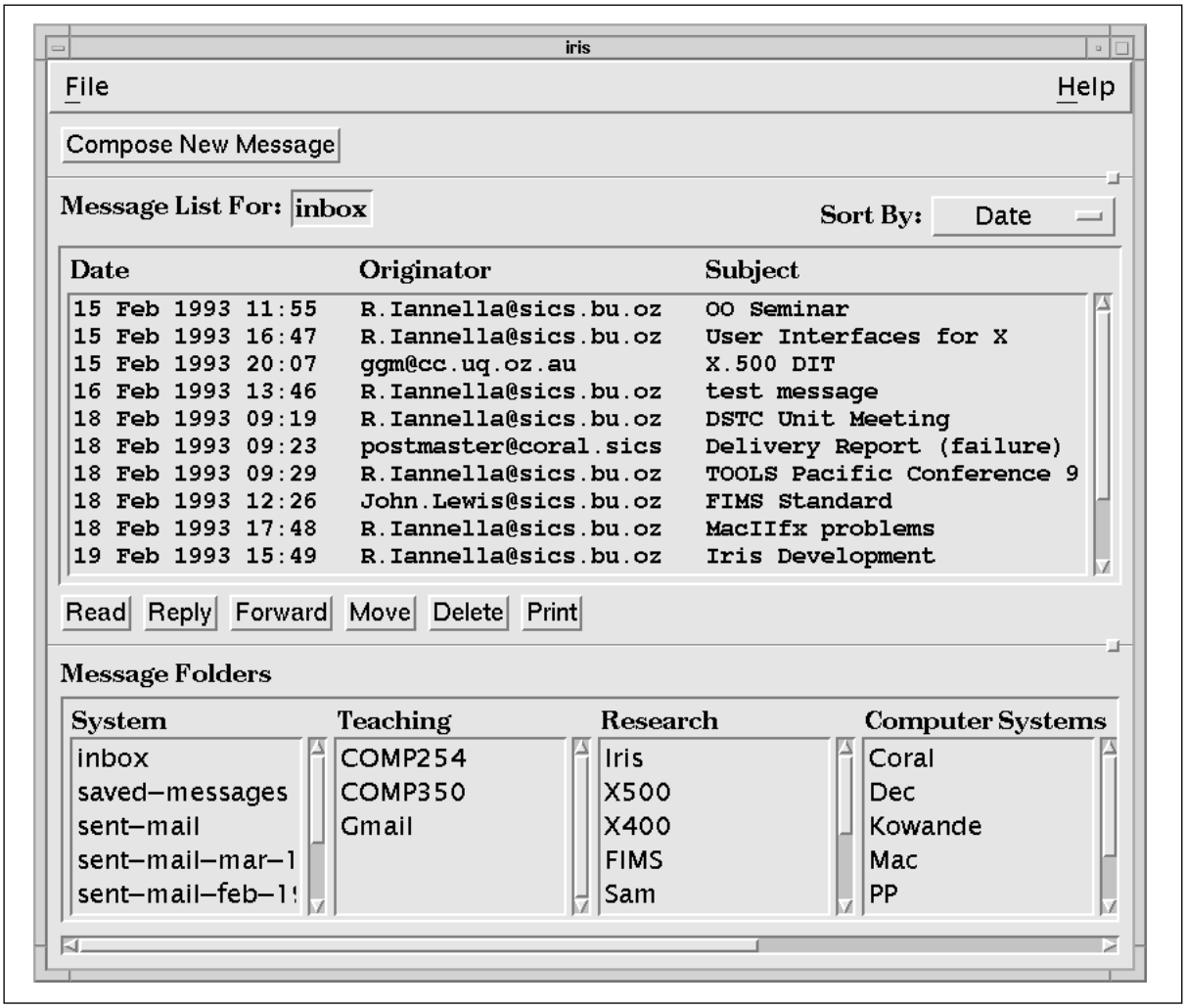


Figure 93: Iris Redesign—Main Window

Figure 94: Iris Redesign—Select Message Dialog

enabled all message folder to viewed and selected in one step and addresses interface problem number 1–4. Figure 95 shows the redesigned move message dialog.

The Send Message window required major effort in redesign as it proved to be the critical screen for the functionality of the Iris UA. Some small changes were made to the names of the New Recipient Address button (changed to 'Enter X.400 Address') and the Search Recipient Address button (changed to 'Lookup Name in Global Directory') to be more specific with each button's functionality. Also, the label 'New Nickname:' was added to indicate that new nicknames would be created with these two buttons. This change addresses interface problem numbers 4–7 and 5–1. Figure 96 shows the new redesigned Send Message window.

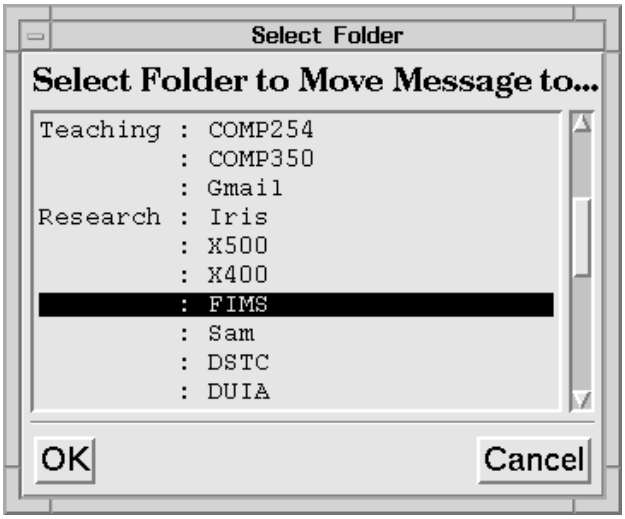


Figure 95: Iris Redesign—Move Message Dialog

The recipient category toggle buttons were moved to be above the 'Add' and 'Move' buttons to give them more prominence. The label 'Add as:' was also included to provide feedback as to the category semantics. This change addresses interface problem number 3–4.

A small change was also made to align the message subject text field with the subject label for layout aesthetics.

The largest change to the send message window was the complete redesign and review of the message and recipient option functions. A major problem seemed to be in the separation of the two option buttons as well as the physical isolation of the Message Options button on the top of the screen. The solution was to classify all the options into four distinct and cohesive groups:

1 Urgency options,
2 Notification options,
3 Date & Time options, and
4 Miscellaneous options.

These message options were then presented as a series of four pushbuttons above the message subject area. These changes addresses interface problem number 3–5.

The Urgency message options consisted of the IPM Importance and Sensitivity options and were a series of toggle buttons for each. Figure 97 shows the redesigned Iris Message Urgency options.

The Notification options for messages are similar to the Recipient options of the original Iris design and include the:

- non-receipt notification,
- receipt-notification,
- return message body, and
- reply requested options.

The Notification options button cannot be selected unless at least one recipient has been moved to the Recipient list. If the user attempts to select this button an appropriate warning dialog is presented (see Figure 98) which addresses interface problem number 3–3.

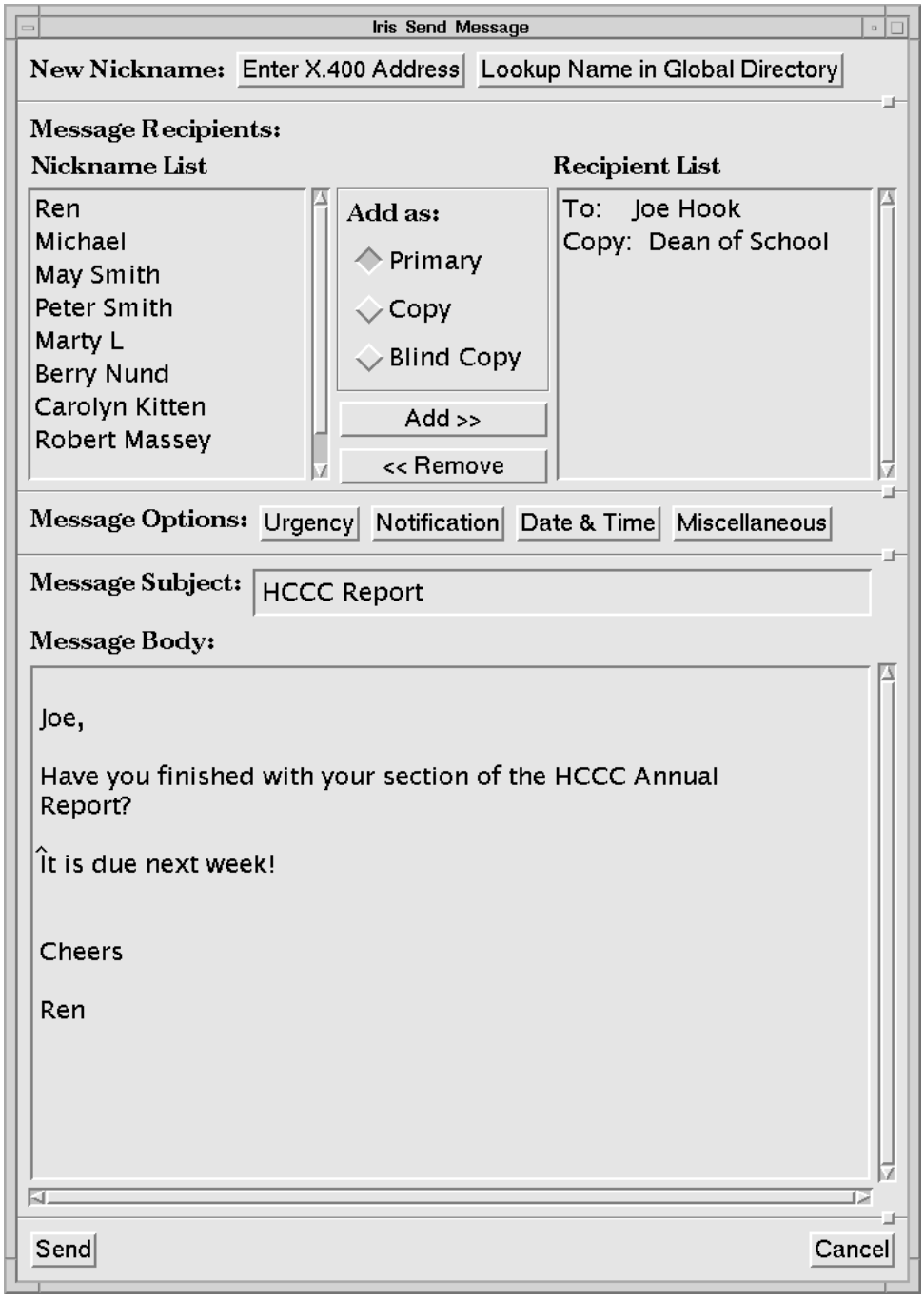


Figure 96: Iris Redesign—Send Message Window

In the case of multiple message recipients, the message Notification dialog includes an option menu that enables the user to select which recipient the notification options apply to. The user may set the notification options for multiple recipients at the same time by simply selecting each recipient from the option menu. As an added function, the option menu contains an alternative called 'All Recipients' which enables the same Notification options to apply to all the message recipients. Figure 99 shows the message Notification dialog.

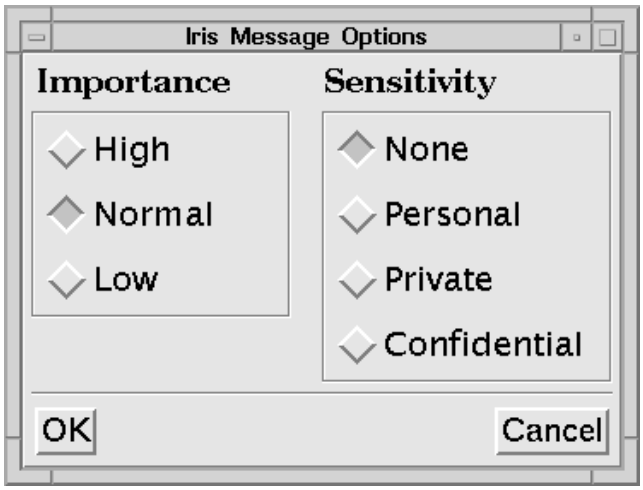


Figure 97: Iris Redesign—Message Urgency Options Dialog

Figure 98: Iris Redesign—No Recipients Selected Warning Dialog

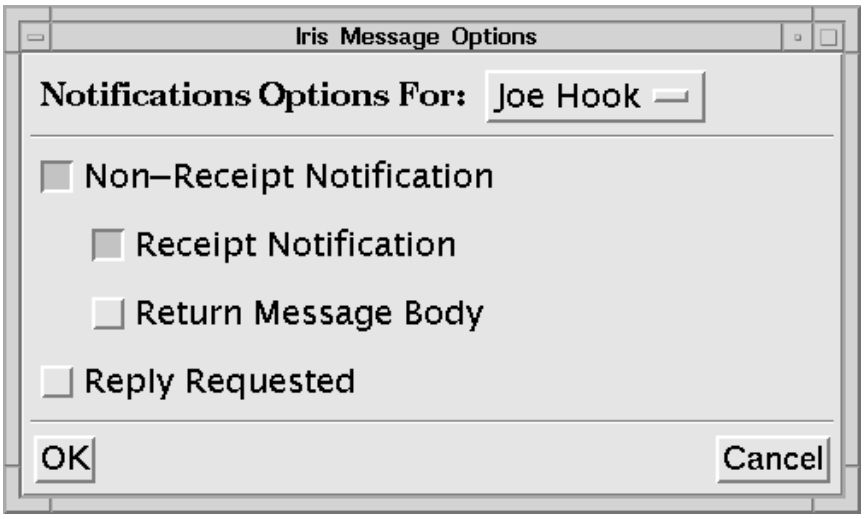


Figure 99: Iris Redesign—Message Notification Options Dialog

The message Date and Time options dialog includes all the options that require a date and time to be specified including the:

- deferred delivery,
- message expiration, and
- message reply-by times.

In the original Iris design, the date and time options were specified by a series of toggle buttons that, when activated, would display a separate dialog window. The user would then en-

ter the date and time through a series of option menus. Figure 100 shows the new redesign of the message Date and Time options dialog. In this dialog, each option is still activated with a toggle button but the date and time option menus are constantly displayed alongside each option label. If the option is off, then all the date and time option menus are insensitive, that is, they cannot be active unless the user has tuned the option on via the toggle button. The exam-

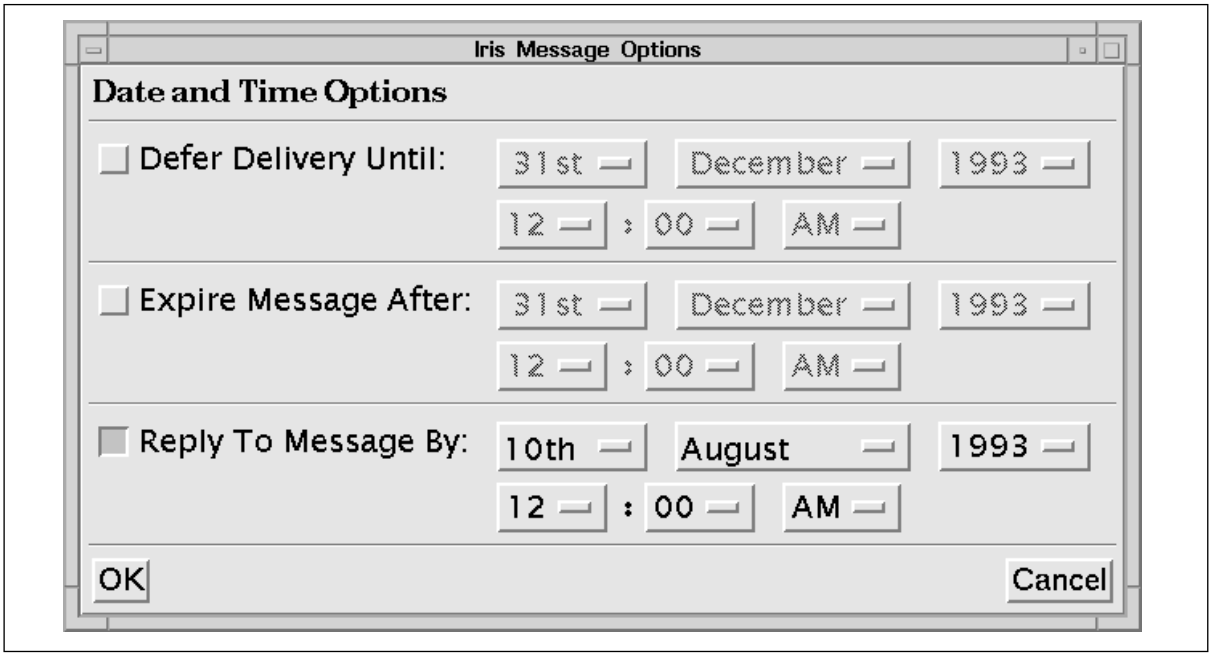


Figure 100: Iris Redesign—Message Date & Time Options Dialog

ple in Figure 100 shows the 'Reply To Message By' toggle button on, with the date and time option menus active. The other two toggle buttons are off, with the corresponding date and time option menus inactive.

The message Miscellaneous options dialog includes options for setting the:

- authorising user,
- reply recipients,
- obsoleted messages, and
- related messages.

Figure 101 shows the redesigned message Miscellaneous options dialog.

The first usability trial identified problems with the O/R address entry screen and its redesign was substantial. The major aspect of the new design is that the user can now tailor or adapt the interface to the requirements matching the supplied O/R address. Previously, the user was presented will all possible O/R address attributes to enter which may have led to erroneous data being supplied, or more simply, confused the user.

The new O/R address screen is now split into two areas. The top half presents a toggle button list of all the O/R address attributes (except the mandatory attributes). The bottom half presents the attribute names and text fields for the entry of the data. By default, the only attributes present in this half are Surname, PRMD, ADMD, and Country option menu. The latter three, being mandatory attributes, and Surname are the default since the majority of O/R addresses would be composed of these as the minimum attributes.

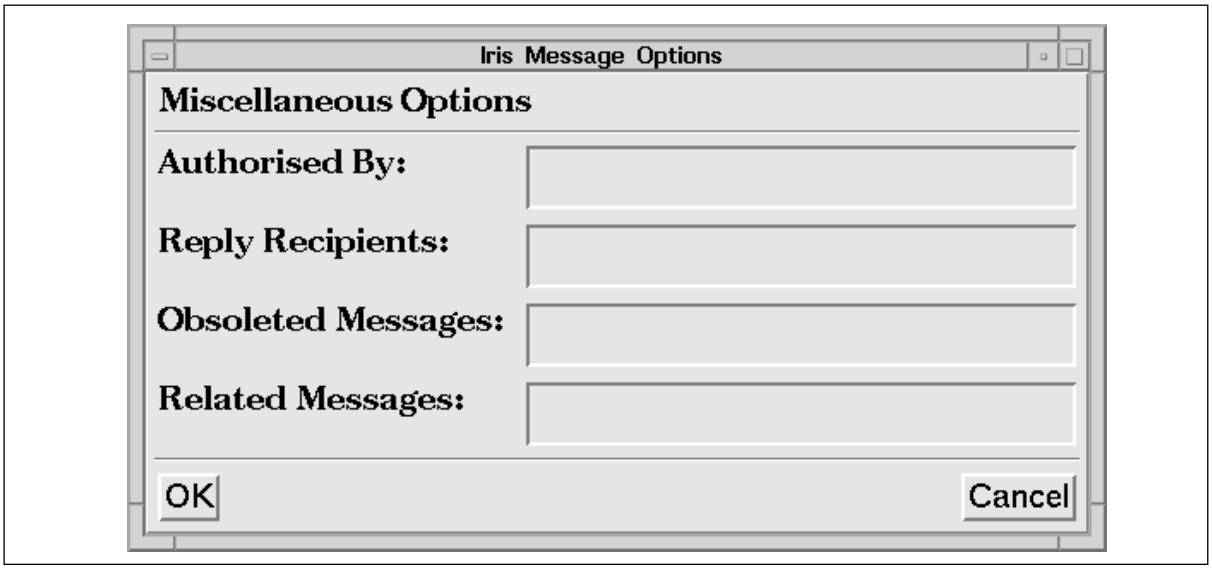


Figure 101: Iris Redesign—Message Miscellaneous Options Dialog

The key to the usage of this form filling screen is that the user can customise the attributes listed in the bottom half of the screen by selecting (or de-selecting) the appropriate toggle button in the top half of the screen. As each attribute toggle button is selected, the attribute name and text entry field will appear (or disappear) from the bottom half of the screen. The position is consistent with the common hierarchy of the attributes. Thus, this process enables the user to match precisely the O/R address that needs to be entered and the screen layout of O/R address attributes. These quite substantial changes were necessary to address interface problem numbers 4–3 and 4–4.

The acronyms used for the O/R address attributes were also reviewed as these were a potential cause for concern. In the redesigned screen, the attribute toggle buttons list the common acronyms after the attribute name so the user can match the two when selecting the required attribute fields. The attribute acronym is displayed to the immediate left of the text entry field with an equal sign between the two. The reason for this is that most textual forms of O/R addresses will use the common 'attribute-acronym = data-value' format. These changes were to address interface problem number 4–5. Figure 102 shows the new redesigned O/R address entry screen.

Another problem with the original O/R address entry screen was that the user occasionally failed to enter a nickname into the top text field on the screen. In the new design, after the user selects the OK button, a dialog appears asking the user to enter a nickname for the newly entered O/R address. This change addresses interface problem number 4–1. Figure 103 shows the nickname entry dialog.

In the redesign of the X.500 directory searching dialogs, only a few cosmetic changes were made. Firstly, some user feedback was added to the top of the main lookup screen to indicate that the details of the person to search should be entered. This is shown in Figure 104.

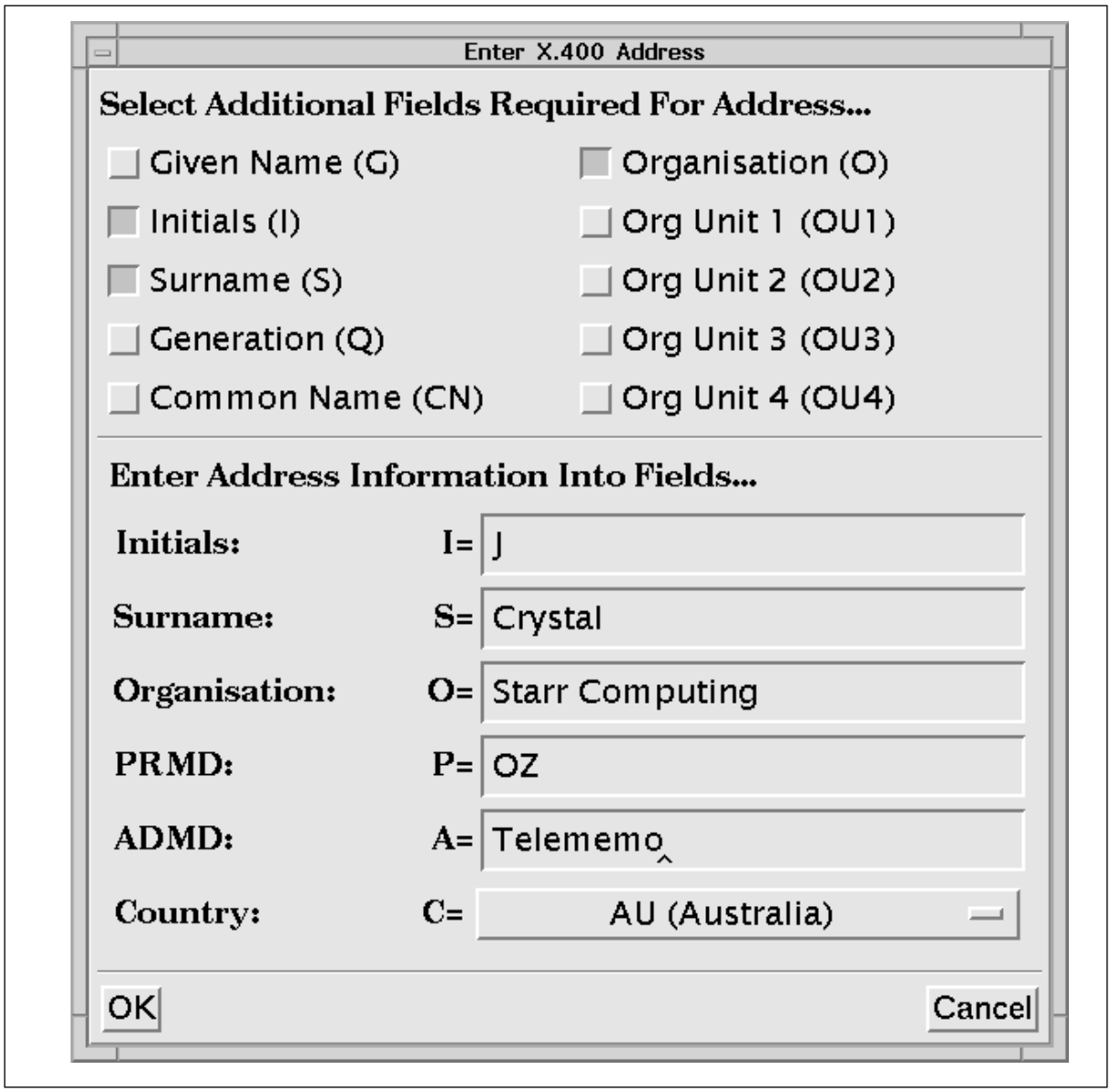


Figure 102: Iris Redesign—O/R Address Entry Screen

Figure 103: Iris Redesign—Enter Nickname Dialog

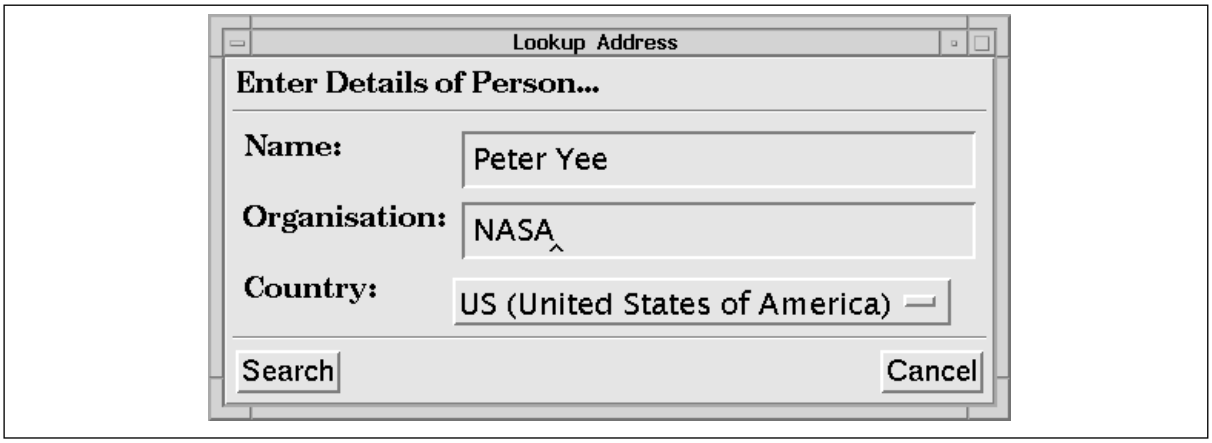


Figure 104: Iris Redesign—Lookup Address Dialog

Secondly, the dialog which presents the user with a list of matches from a search was changed in three ways:

1 The wording at the top of the dialog was changed to ask the user to select from the following matching entries (interface problem number 5–4).

2 Feedback on the meaning of the attribute acronyms is displayed on the bottom of the dialog (interface problem number 5–5).

3 The 'Continue Search' button was changed to 'OK' (interface problem number 5–3).

These changes to the directory lookup dialog can be seen in Figure 105.

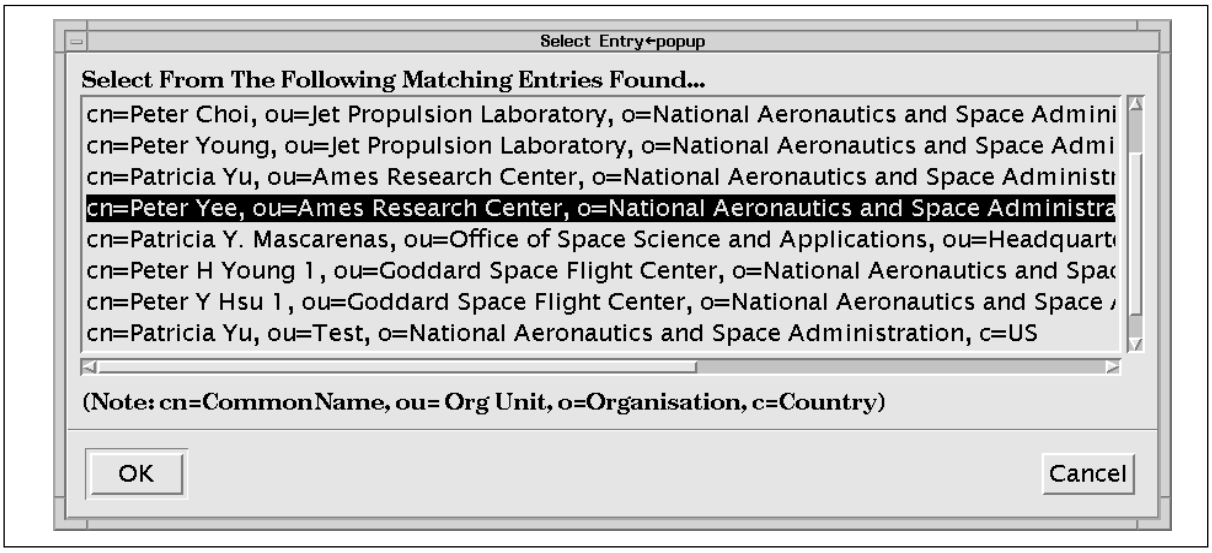


Figure 105: Iris Redesign—Lookup Matches Dialog

Finally, one small change was made to ensure that a user enters a nickname when presented with the Enter Nickname dialog box. This was to stop null nicknames been accepted by the prototype and addresses interface problem number 5–2. Figure 106 shows the nickname entry warning dialog.

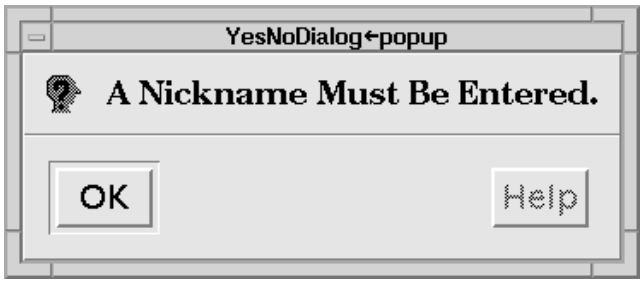


Figure 106: Iris Redesign—Nickname Entry Warning Dialog

6.3.2 Scenario Business Card Redesign

The final problem identified in the first usability trial was the layout of the X.400 address on the business card in task four (interface problem number 4–2). Some trialists had problems in firstly identifying that the X.400 address existed on the card, and secondly, in interpreting the syntax of the address. To solve the former problem, the new business card design dedicated more space to the X.400 address. (The back of business cards may have to be used in some cases to support this.)

To solve the latter problem, the X.400 address is presented in a vertical format with each attribute/value pair listed in the familiar 'attribute-acronym = data-value' format. This is also consistent with typical 'form-fill' screen layouts. The original business card displayed the attribute/value pairs horizontally over two lines. The other advantage to this format is that it directly mimics the format of the redesigned O/R address entry screen, thus enabling the user to map the two representations. Figure 107 shows the redesigned business card used in task four.

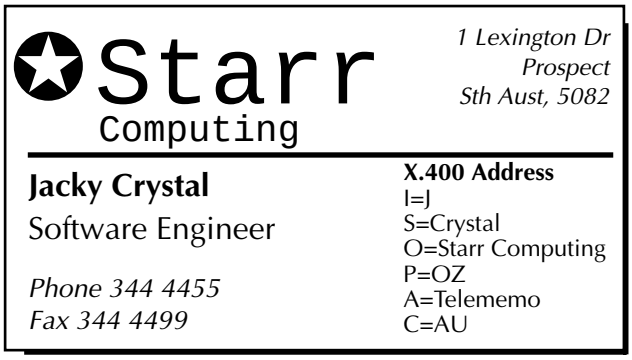


Figure 107: Scenario Business Card Redesign

On key issue with the change to the layout of the O/R address on the business card is that the task is still unchanged. That is, since a comparison will be made across the two groups of trialists, it is important to note that both will still be subjected to the same task requirements. Of course, since this research is investigating user interfaces, the interface of the business card has changed which is the same situation with the interface of the Iris UA prototype. Hence, the modified business card layout, which is necessary for this research, still preserves the original task semantics and will allow comparative evaluations.

6.4 Usability Trial Two Results

With all the previously mentioned changes applied and tested in the new Iris UA prototype, the second usability trial was performed using a different set of trialists. The same experimental method was employed to administer the trials and to analyse the video tapes. In this trial, noticeable improvements were achieved by the group of trialists. In particular, the majority of redesigned user interface aspects proved to be successful in providing the appropriate functionality and user interaction sequences. This was clearly indicated by the results from the questionnaires and task times. The subjective questionnaires showed a marked increase for the second trialists. The task-completion times proved conclusively that the new redesigned Iris prototype had significantly decreased the time required to complete each task. A complete analysis is discussed in section '6.5 Questionnaire Results' on page 162 and section '6.6 Task Completion Time Results' on page 165.

The review of the video tapes did highlight a few recurring problems:

1 When required to send a new message, a few trialist were still unable to see the 'Compose New Message' button at the top of the screen.
2 In task three, a few trialists added the 'Dean of School' as a primary recipient instead of copy recipient.
3 When required to address a message to a new recipient in task four, some trialist used the Directory Lookup screen instead of the O/R Address form screen.

It was unclear why these three problems occurred. The wording of the tasks may have made it difficult to understand the exact requirements. In the first case, the videos showed evidence that the trialists were sometimes reluctant to look at the top of the screen (where the 'Compose New Message' button is located). Instead, they concentrated on the message list and folders section of the screen. This may indicate that the button may need to be moved.

In the second case, the trialist's interpretation of 'send a copy to the Dean of the School' from task three may have been too imprecise. In the third case, the 'mistake' will in fact be the preferred method of addressing recipients. Hence, the trialist's error would not be harmful and may even save the entry of the O/R address if the recipient is found in the directory.

Overall, the second trial proved to be very successful with the increase of usability evident from the review of the video tapes. The design changes made to the interface after the first trial identified and solved the original problems. The improvements in user performance (ie less hesitation, errors, uncertainty, and redundant moves and selections) was indicated by the little or no problems faced in the second trial. This suggests improved usability along the 'effectiveness' dimension. The problems found after the second trial, listed above, did not warrant another redesign and third usability trial.

One interesting point to the two trials is that no user attempted to use the on-line help facility at any stage, even when they seemed in need of assistance. The help menu is located on the top right of the main window, and although it did not actually contain any useful help messages, it was still surprising that users did not utilise this function. Instead, the users opted to 'battle' with the interface when faced with problems. The users may have either been unaware that this facility existed or did not access the on-line help as the tasks did not require this activity. Alternatively, this may imply that the users were not familiar with the availability of on-line help in electronic mail systems. (Unfortunately this is often the case, particularly for Unix electronic mail systems.)

6.5 Questionnaire Results

An analysis of the questionnaires comparing the two different groups of trialists, before and after the interface changes to Iris, added further evidence to the successful design changes. Each rating in the interface and functionality questionnaires were assigned values between 1 (lowest) and 5 (highest). A statistical analysis of the data collated from the questionnaires was performed to establish if any significant differences existed between the two groups of trialists. The functionality and interface questionnaires were further summarised to reflect high level aspects of each questionnaire.

The details of the two sample differences test using both a parametric (T-Test) and a non-parametric test (Wilcoxon) are listed in Appendix N and summarised below. The T-Test used was a one-tailed analysis for matched samples since the data was expecting the differences to be in one particular direction (ie questionnaire ratings to increase and task times to decrease). The T-Test for the functionality and interface questionnaires had 14 degrees of freedom and 4 degrees of freedom was used in the task completion time analysis. The Wilcoxon test used a matched samples analysis.

The basic test used was the Wilcoxon test since the data clearly did not come from a normal distribution being ratings from one to five. Often non-parametric tests do not detect differences too well, so it was helpful to be able to use a parametric procedure to confirm the findings. (However, the Wilcoxon test did clearly show the differences.) The T-Test was used since this procedure is fairly robust to departures from the normal distribution, other than skewness and outliers, and since low numbers of trialists were used. There were no outliers in the data because of the limited range and taking differences means that skewness is not likely to be of concern.

The results suggest that the second group of trialists liked the redesigned Iris UA prototype more than the first group. However, since the questionnaires are subjective, they refer to only one dimension of usability (ie attitude).

6.5.1 Experience Questionnaire

The user experience questionnaire confirmed that the majority of trialists were familiar with some form of text- or graphical-based electronic mail system. None were seen to be classified as ‘power-users’ of messaging systems. The other key point of the user experience questionnaire was that it reconfirmed that all trialist had no previous knowledge of X.400 message handling systems. A summary of the user experience questionnaires for both trials can be found in Appendix H.

Although randomly selected, the second group of trialist indicated that, overall, they had greater ‘experience with computers’ than the first group. All of the second group stated greater than three years experience with 33% of the first group stating the same. This may have led to some bias in the findings. However, the first group indicated an advantage in the ‘experience with graphical interface electronic mail systems’ (50% less than two years, 50% less than one year) over the second group (83% less than one year, 17% no experience). These two questionnaire findings, taken together, indicate that the two groups of trialists may effectively be compatible for cross-comparisons.

6.5.2 Functionality Questionnaire

The functionality questionnaire is summarised in Table 27 and shows an increase in the average ratings for all the identified functional areas. The averages show a marked increase in problem areas redesigned after the first trial. These include:

- sending a mail message,
- recipient options, and
- all aspects of message addressing.

The average ratings show a significant difference in the two trials in both the T-Test [$t(14) = 5.99$, $p < 0.0001$] and Wilcoxon test [$p < 0.001$]. The full results of the functionality questionnaire for the first usability trial is listed in Appendix I. Appendix J lists the results for the second usability trial.

Table 27: Functionality Questionnaire Summary

Functional Area	Question	Trial One Average	Trial Two Average
Message Management	read a mail message	4.3	4.5
	reply to a mail message	4.3	4.7
	send a mail message	2.8	4.2
	move a mail message to a folder	3.8	4.5
	move to different mail folders	3.8	3.7
	delete mail messages	4.3	5.0
	print mail messages	4.3	5.0
	Average	4.0	4.5
Options	message options	3.2	3.7
	recipient options	2.3	4.0
	Average	2.8	3.8
Addressing	using the nicknames	3.2	4.2
	given an X.400 address	2.2	3.7
	using directory searching	3.3	4.2
	Average	2.9	4.0
Overall	adequate functions for electronic mail	4.5	4.8
	easy access to all options for electronic mail	3.5	4.5
	support for appropriate electronic mail tasks	3.8	4.8
	Average	3.9	4.7

The functionality questionnaire was also summarised into the major functional areas of the Iris UA. The average values for these new areas were extrapolated from the data corresponding to each relevant question (from column one in Table 27). Figure 108 graphically shows the clear increases in the average ratings for the high level functionality of the Iris UA.

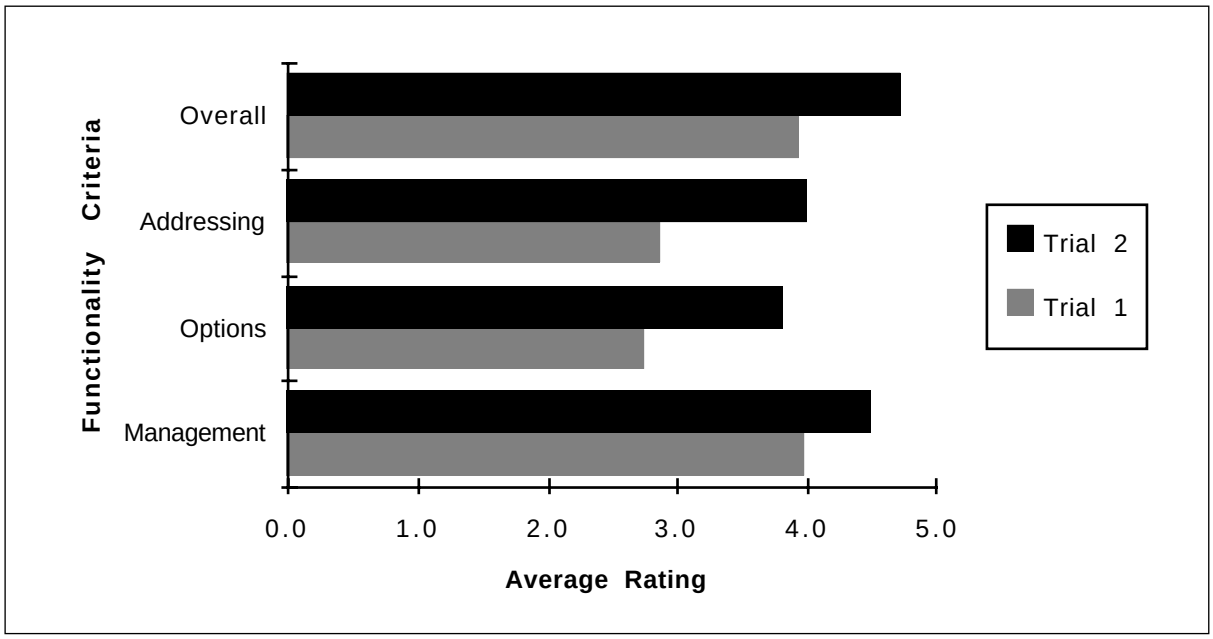


Figure 108: Functionality Summary Chart

6.5.3 Interface Questionnaire

The graphical user interface questionnaire is summarised in Table 28 and shows an increase in the average ratings for all the identified interface areas. The exception is the third question that addresses the consistency issue. This may be attributed to the problem identified after the second trial in the location of the 'Compose New Message' button. This button may have been better located with the message list buttons. This way, all functional buttons would have located in the same position. The one argument against this change would be that the 'Compose New Message' button is not related to any of the functions for existing messages.

The average ratings show a significant difference in the two trials in both the T-Test [$t(14) = 6.44$, $p < 0.0001$] and Wilcoxon test [$p < 0.001$]. The full results of the interface questionnaire for the first usability trial is listed in Appendix K. Appendix L lists the results for the second usability trial.

Table 28: Interface Questionnaire Summary

Interface Area	Question	Trial One Average	Trial Two Average
Visual Clarity	Does the information appear to be organised logically on the screen	3.8	4.2
	It is easy to find the required information on the screen	3.3	3.7
	Average	3.6	3.9
Consistency	Is the information (eg menus, buttons) displayed in a consistent location and layout	4.5	4.2
	Is the way the system respond to a user actions consistent at all times	3.8	4.3
	Average	4.2	4.3

Table 28: Interface Questionnaire Summary *(continued)*

Interface Area	Question	Trial One Average	Trial Two Average
Expectations	Is the format of displayed information compatible with the users perception	3.2	4.2
	Does the sequence of activities required to complete a task follow what the users would expect	3.7	4.3
	Average	3.4	4.3
Feedback	Are messages and instructions relevant and clear to the user	3.7	4.2
	Are status messages informative and accurate	3.3	4.5
	Average	3.5	4.3
Explicitness	Where the user is presented with a list of options, is it clear as to what each option means	3.5	4.0
	Is it clear what different parts of the system do	3.3	4.0
	Average	3.4	4.0
Flexibility	Do users have control over the order of information entry	4.0	4.5
	Is the system flexible in allowing the user to choose options	4.3	4.7
	Average	4.2	4.6
Error Prevention	Does the system allow the user to easily correct data	3.2	3.8
	Does the system protect against errors in user actions	3.2	3.8
	Average	3.2	3.8
Overall	Overall, the interface was pleasing and easy to use	3.7	4.2

The interface questionnaire was also summarised into the major aspects of the Iris UA interface. The average values for these new areas were extrapolated from the data corresponding to each relevant question (from column one in Table 28). Figure 109 graphically shows the clear increases in the average ratings for the high level graphical user interface aspects of the Iris UA.

6.6 Task Completion Time Results

The average time taken for each task in the two usability trials is graphically summarised in Figure 110. The average times show a significant difference in the two trials in both the T-Test [$t(4) = -8.71$, $p = 0.0005$] and Wilcoxon test ($p = 0.030$). Substantial savings were made (over two minutes in some cases) in the completion times of the tasks in the second trial. The full results are listed in Appendix M for each task and trialists.

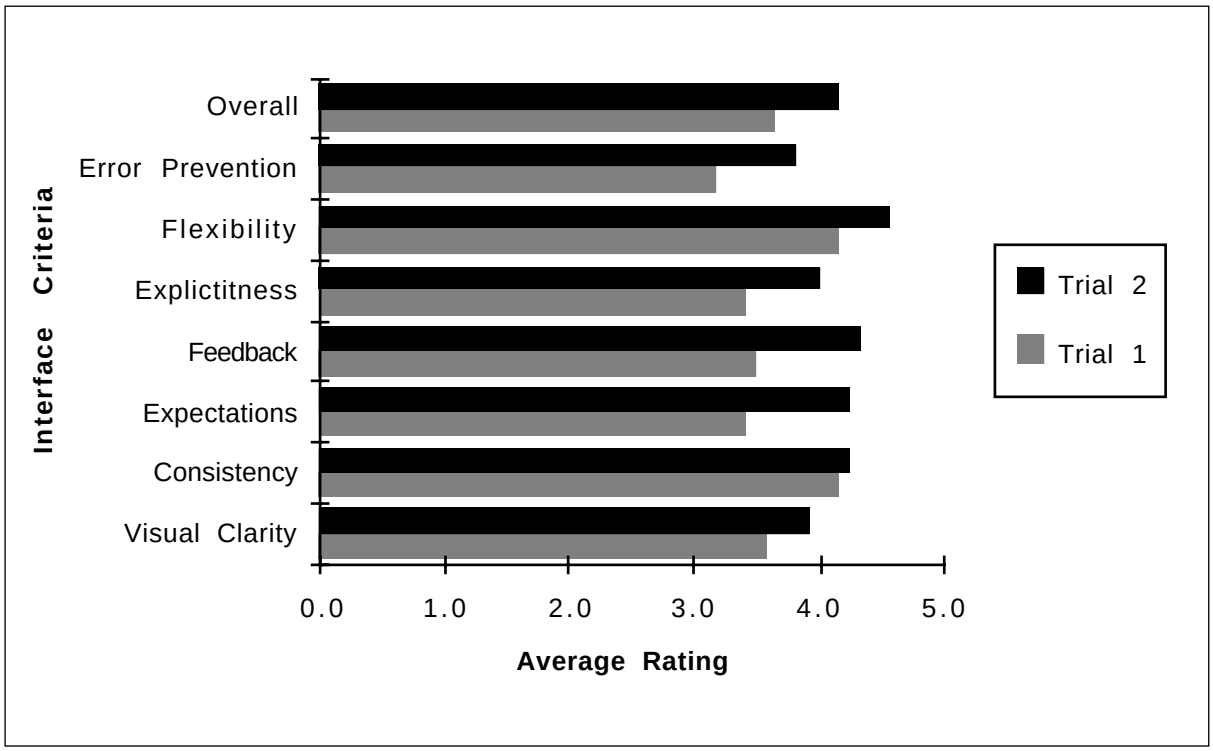


Figure 109: Interface Summary Chart

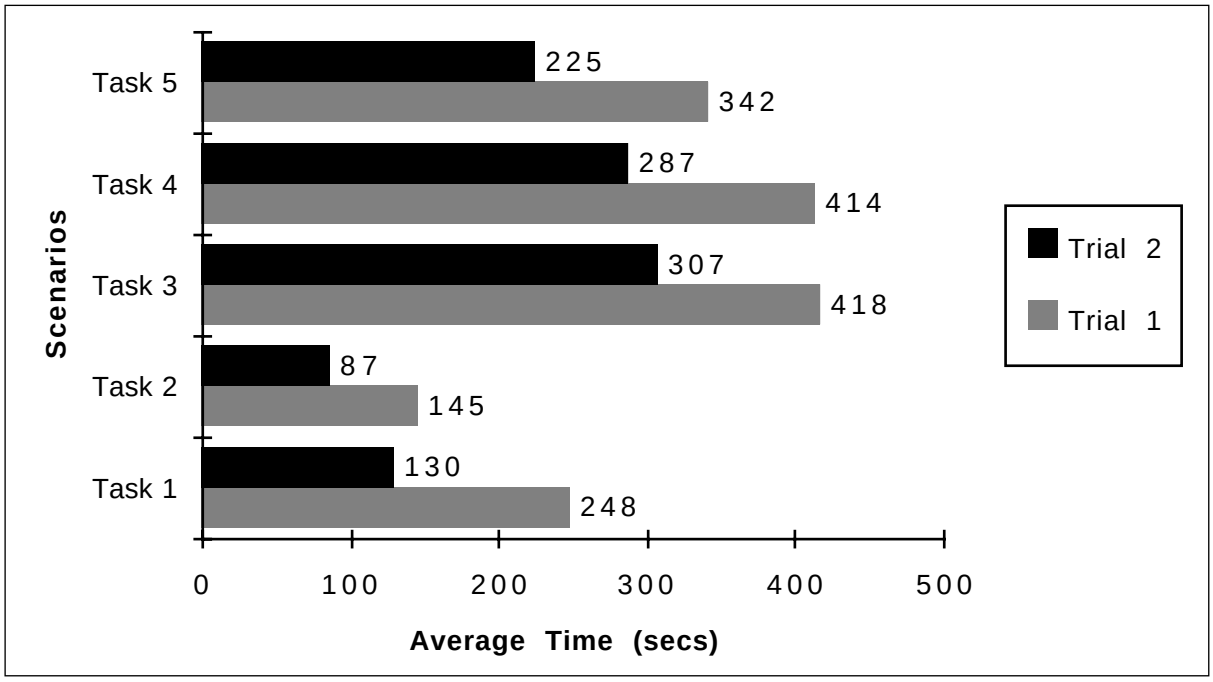


Figure 110: Task Time Summary Chart

6.7 Summary

The usability evaluation proved very successful in identifying the interface problems of the Iris UA. The use of scenario-based tasks was an effective method for the video taped evaluation sessions. The analysis of the videos allowed the dissection of each trialist's interaction with the software. The results enabled a focused redesign of the interface for Iris. The iterative process continually narrowed the user interface problem areas. As the final video analysis

showed, the user performance had improved with little or no indication of any major usability problems.

Although subjective, the questionnaire results added further weight to the successful interface changes by showing an improved attitude by the second group of trialists. However, the major findings were reflected in the task-completion times. The second group of trialists exhibited a substantial reduction in the task times. This conclusively shows that the graphical user interface enhancements made to the Iris UA were effective in increasing its usability.

Overall, the choice of usability methods enabled a comprehensive evaluation and redesign of the Iris UA. The results of which indicated that a productive and functional graphical user interface for the Iris UA had been reached.

Chapter 7

Reference Model

7.1 Introduction

The reference model describes the graphical user interface requirements for X.400 IPM user agents. The reference model resulted from the design, development, and evaluation of the Iris UA interface that has been outlined in Chapters five and six. The Iris Reference Model (IRM) is based on industry standard user interface guidelines including novel, adaptive, and extensible interaction techniques. The IRM is also based on proven standards for electronic messaging application domains.

In developing the IRM, it must be noted that the model reflects the features of the Iris UA prototype that were found to be the most effective. However, some alternatives that were not considered in the design of the Iris UA prototype cannot be totally dismissed. For example, the three dimensional 'perspective wall' (Mackinlay *et al*, 1991) may offer other alternatives to interacting with messages. Unfortunately, such systems were technically infeasible to implement with the development environment used for the Iris UA prototype.

The IRM establishes general software engineering principles and practices for the design and development of X.400 UA systems and facilities. The purpose is to present applicable design criteria to achieve a satisfactory level of performance by IPM users. This will minimise the skill and training requirements for such users and also achieve a reliable degree of competency. The IRM is designed to foster standardisation within and among the messaging applications domain.

7.2 Iris Reference Model

The IRM is presented as a complete standard for IPM UAs. Portions may be used separately for similar messaging applications. The IRM assumes a number of minimal capabilities for the graphical user interface environment. These include minimum, maximum, and default values for each user interface object. The major assumption relates to the user's ability to resize interface objects dynamically. For example, the interface should support the user's ability to resize a scrolling list of nicknames to display more or less items. (This feature is available in the

OSF/Motif environment with the use of paned windows.) If unable to support this requirement, other environments should closely follow the recommendations for dynamic options. Specification for the sizes for text entry fields should follow the recommendations given in the X.420 standard for each option or attribute.

The IRM recommendations are based on the experimental evidence of the usability trials of the Iris UA, the Smith & Mosier guidelines, existing messaging applications, as well as user interface design experience of a range of messaging systems. The justification of each recommendation has been extrapolated from this vast source of data. The IRM is categorised into eight sections with each section describing the recommendations of a specific aspect of UAs. These areas are each an essential requirement for the provision of messaging system services. These areas have also been identified in the X.420 standard as the functional requirements of IPM user agents. The eight sections of the IRM include:

1 message lists
2 message folders
3 reading messages
4 sending messages
5 message options
6 message addressing
7 directory integration
8 user agent administration

A summary of the recommendations is provided at the end of each section. The complete list of all IRM recommendations can be found in Appendix O.

Throughout this chapter, the use of the words 'shall', 'should', 'may', and 'will' in this model is in accordance with MIL–STD–962[1], wherein 'shall' expresses a provision that is binding, 'should' and 'may' express nonmandatory provisions, and 'will' expresses a declaration of future purpose.

7.2.1 Message Lists

The message list shall be located in a highly visible section of the UA window as it will provide the main interaction sequences with the messaging system. This should be the top half of the main window. Each message shall be displayed with a one-line summary including the message date, originator, and subject. Where supported, an indication of the urgency of the message should also be included.

Each message line can be selected and a number of activities applied to it. Such operations include; read, reply, forward, move, print, and delete. The default action, provided by a double-click selection of the message line, shall be to read the message (see '7.2.3 Reading Messages' on page 172). If no message is selected, then none of the options can be selected, and will generate a warning to the user.

Selecting the reply option shall generate a new message, pre-addressed to the message originator and any specified reply recipients. The subject field shall be the original message subject preceded with 'RE: '. The forward option shall generate a new message including the original message body text surrounded by '——Start Forwarded Message——' and '——End

[1] US Department of Defense Military Standard.

Forwarded Message——' indicators. The subject field shall be the original message subject preceded with 'FWD: '. For both replies and fowarded messages, the 'RE: ' and 'FWD: ' text should only be added if this is the first occurrence of such an operation. Both the delete and print options should confirm their action before proceeding.

The move option shall provide a scrolling list of all the message folders for the user to choose from. The list shall be a one-dimensional representation of the message folders hierarchy. Once selected, the message shall be moved to the new folder location. If the platform supports a drag-n-drop interface, then each message should be capable of being dragged to the destination folder. In this case, each line of the message list shall be preceded with a small generic document icon and the message folders with a small generic folder icon (see '7.2.2 Message Folders').

The message list shall also display the name of the currently viewed folder. The default folder name for new messages should be 'inbox'. The messages in the message list shall also be able to be sorted by date, originator, subject, or priority. The date should be displayed in the default format applicable to the operating system. The originator's name should be displayed, if possible, rather than the O/R address together with the subject line. The message operations shall be displayed under the message list as a series of buttons.

Figure 111 shows an example of an IRM message list. A minimum of five messages should be displayed by default.

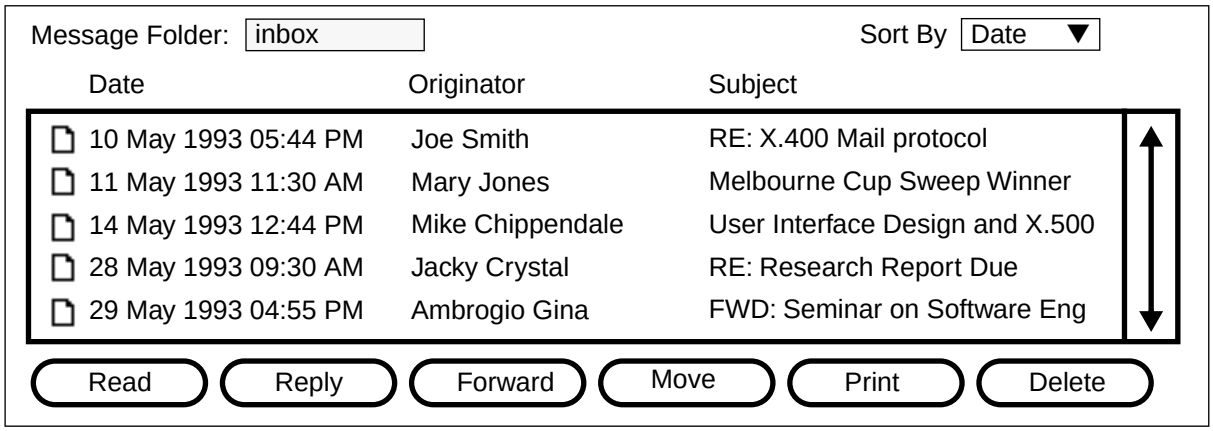


Figure 111: IRM—Message List Example

Table 29 summarises the IRM message list recommendations.

Table 29: IRM—Message List Recommendations

Number	IRM Recommendation
1/1	Message list displayed in prominent section of the main window
1/2	One-line summary of each message (date, originator, subject) displayed in the message list. Message urgency should also be displayed. Date format determined by the default operating system. Originator's name to be displayed rather than the O/R address.
1/3	Read, reply, forward, move, print, and delete functions to be supported for each message and displayed under the message list. Default function is to read a message (see IRM–3/1).

Table 29: IRM—Message List Recommendations *(continued)*

Number	IRM Recommendation
1/4	Reply function generates a new message (see IRM–4/1). The message is pre-addressed to the originator and any reply-recipients. Message subject preceded with 'RE: '.
1/5	Forward function generates a new message (see IRM–4/1). Body of original message text is included in the forwarded message surrounded by indicators. Message subject preceded with 'FWD: '.
1/6	Print and delete functions confirm their actions.
1/7	Move option displays a list of all message folders (a one-dimensional representation) to move message to. If drag-n-drop environment supported, then message icon can be dragged over message folder icon (see IRM–2/5).
1/8	Current message folder name displayed above the message list.
1/9	The default folder name for new messages is 'inbox' (see IRM–2/7).
1/10	Message list can be sorted by date, originator, subject, or priority.
1/11	Message list should display a minimum of five messages.

7.2.2 Message Folders

The message folders provide a method to store and retrieve existing messages. Management facilities to add, remove, and rename message folders shall be provided in a convenient manner. Since a single list of message folders will become overwhelming long over a short period of time, the user shall have the ability to group similar message folders together. Each message folder group will itself be a list of message folders, thus providing a two-dimensional view of the folders.

To change message folders (and hence, the corresponding message list display) the user shall be able to double-click a message folder name. To move messages to message folders, a single scrolling list shall be provided for the user to select from. The single list should represent the two-dimensional folder messages by listing each group name with the individual subfolder names. The subfolder name should be indented appropriately to convey the two-dimensional structure. If the platform supports a drag-n-drop interface, then each message folder shall be preceded with a small generic folder icon enabling the message items to be dragged over it for filing.

By default, one message folder group called 'System' shall be created. The System group shall contain folders that are required for the use of IPM services. All incoming messages, either new or read and unfiled, shall be stored in the 'inbox' folder. The System folder shall also be used for the 'drafts' message folder. The UA should provide facilities to automatically save outgoing messages. In this case, the messages should be stored in the 'sent-mail' folder. Each month, the user should be asked to save the sent-mail folder messages into a new date-specified folder for archiving (eg 'sent-mail-jan-1993', 'sent-mail-feb-1993', etc).

Figure 112 shows an example of an IRM message folder configuration. By default, four folder groups should be displayed each with a minimum of five folders listed.

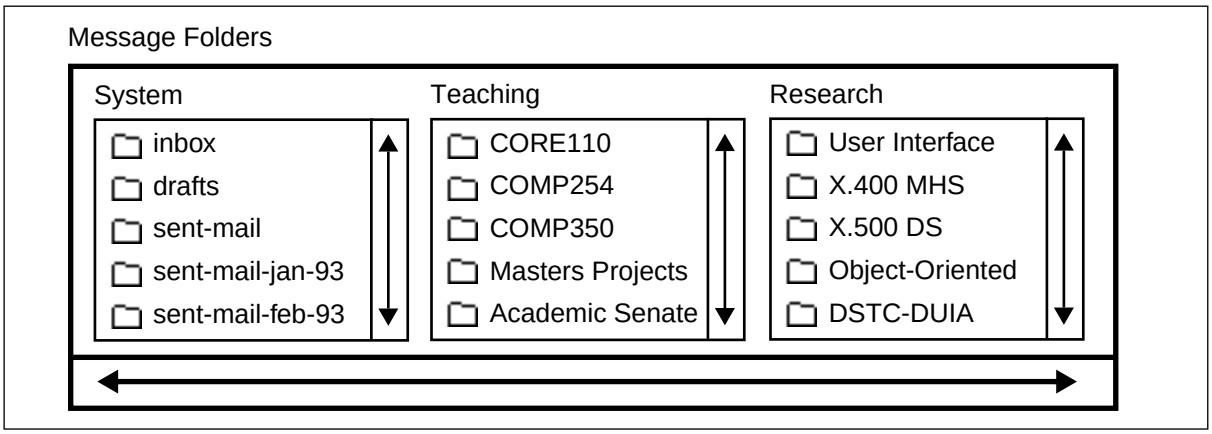


Figure 112: IRM—Message Folders Example

Table 30 summarises the IRM message list recommendations.

Table 30: IRM—Message Folder Recommendations

Number	IRM Recommendation
2/1	Message folder management facilities (add, remove, modify) should be provided
2/2	Message folders are associated into distinct groups. Each group being a list of message folders.
2/3	A two-dimensional view of the message folders shall be displayed on the main window in scrolling lists. Each group name is displayed above a scrolling list of the associated folder names.
2/4	Double-clicking on the message folder name will change the list displayed in the message list (see IRM–1/8).
2/5	If drag-n-drop can be supported, then each message folder name is displayed with a generic folder icon. This enables a message to be dragged over the icon for filing.
2/6	A dialog box interface for moving messages to the folders shall also be supported. The dialog will contain a single list of all message folders, indented under their message group names, for user selection.
2/7	A default 'System' message group shall be available. This shall contain the 'inbox' folder (for new messages) and other system-dependent folders (eg drafts and sent-mail).
2/8	The 'sent-mail' folder should automatically, at end of each month, be either removed or saved into another date-specific folder.

7.2.3 Reading Messages

Reading messages shall involve the display of a new window including all the information contained in the IPM. The body of the message shall be displayed in a scrolling text field. The key service elements (heading fields) shall be displayed in separate and distinct fields with ap-

propriate labels. These include originator, subject, and date. Other service elements should be presented in an 'attribute=value' text format in a single scrolling field.

The message operations shall be displayed under the message body as a series of buttons. These include; reply, forward, move, print, and delete. There shall also be a button to close the message window.

Table 31 summarises the IRM reading messages recommendations.

Table 31: IRM—Reading Messages Recommendations

Number	IRM Recommendation
3/1	Reading a message displays a new window with the IPM details shown.
3/2	The message body shall be displayed in a large scrolling text field.
3/3	The key heading fields (originator, subject, date) shall be displayed in separate text fields.
3/4	Other message heading fields should be displayed in a single scrolling field in an 'attribute=value' format.
3/5	Message operations, including reply, forward, move, print, and delete, shall be provided on the message window.

7.2.4 Sending Messages

The composition of new messages shall display a new window for the entry of the details of the message. The new message window should be created from a clearly labelled 'Compose New Message' button on the main window.

The top area of the window should contain mechanisms to specify the recipients of the message. There shall be a main scrolling list of predefined nicknames from which the user may select any number of recipients. Another recipient list shall be present that will display all the recipients selected for the new message. Double-clicking the mouse button on a nickname will move it to the recipient list. Similarly, double-clicking on a recipient name will move it back to the nickname list, and hence, removing the nickname from the message recipients. Two buttons shall also be provided to perform the same functions with single selection of the nicknames. The buttons shall be named 'Add >>' and '<< Remove' with the chevrons indicating the direction of the move. With systems supporting a drag-n-drop interface, the nicknames should also be able to moved with direct manipulation techniques. In this case, the nicknames should be displayed with a small generic icon representing the recipient user. The user should be able to drag the nickname icon over the recipient list for addressing.

By default each nickname shall be added to the recipient list as a primary recipient. There shall be provisions, via a series of radio buttons, also to specify the carbon-copy and blind-copy recipient categories. Once selected, any new additions to the recipient list will be with the new category specification. For example, if the carbon-copy category is currently selected, any subsequent nicknames added to the recipient list will carry that specification. The recipient list shall also indicate the recipient category by preceding the nickname with 'To:', 'Copy:' or 'Blind:' for primary, carbon-copy, and blind-copy respectively.

Figure 113 shows an example of the layout for the recipient specification for new messages. A minimum of ten items should be displayed in the nickname list.

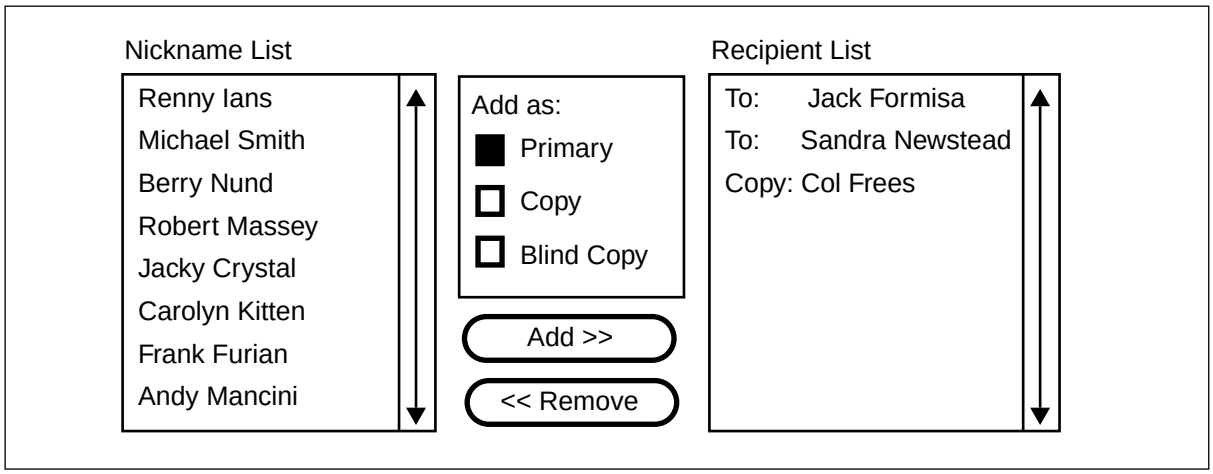


Figure 113: IRM—Recipient Specification Example

Close to the recipient specification area, there shall be two buttons for the creation of new recipient nicknames corresponding to O/R address entry and directory searching. The buttons should be named 'Enter X.400 Address' (see '7.2.6 Message Addressing') and 'Lookup Name in Global Directory' (see '7.2.7 Directory Integration') respectively.

Close to the centre of the new message window should be a series of buttons for the specification of message options. The buttons shall be named 'Urgency', 'Notification', 'Date & Time', and 'Miscellaneous'. Their functions are further discussed in section '7.2.5 Message Options'.

The bottom half of the new message window shall contain areas for the entry of the message subject and body. The message subject shall be a single-line editable text field labelled with 'Message Subject:'. The majority of the new message window shall contain a large scrolling editable text field for the entry of the message body with the 'Message Body:' label displayed at the top of the field.

On the bottom of the new message window shall be two buttons to send the message and to cancel the message composition. The send button will submit the composed message to the appropriate underlying X.400 MTA and provide appropriate feedback as to the progress of this submission process. The cancel button, after confirming this action, will discard the message composition window. A 'draft' button should also be supplied that would file the unfinished message to the draft message folder for later completion and submission.

Table 32 summarises the IRM sending messages recommendations.

Table 32: IRM—Sending Messages Recommendations

Number	IRM Recommendation
4/1	New messages shall create a new window for the entry of the IPM data.
4/2	New messages shall be created from a 'Compose New Message' button on the main window.
4/3	A scrolling list of nicknames shall be available for the specification of message recipients. The message recipients shall be displayed in a separate Recipient list.
4/4	Nicknames can be moved from/to the recipient list via double-clicking on the name.

Table 32: IRM—Sending Messages Recommendations *(continued)*

Number	IRM Recommendation
4/5	Two buttons; 'Add >>' and '<< Remove' shall also allow nicknames to be added/removed from the recipient list with single selection of the nickname. The buttons are located between the nickname and recipient lists.
4/6	Systems supporting drag-n-drop shall allow nicknames to be dragged from/to the nickname and recipients lists. In this case, a small generic icon will be displayed adjacent to each nickname.
4/7	Three recipient categories shall be supported; primary, carbon-copy, and blind-copy. These shall be displayed as a series of toggle buttons between the nickname and recipients lists.
4/8	Additions to the recipient list will display the category specification selected via the toggle buttons. These shall be 'To: ', 'Copy: ', or 'Blind: '.
4/9	Two buttons allow the creation of new nicknames recipients; 'Enter X.400 Address' (see IRM–6/1) and 'Lookup Name in Global Directory' (see IRM–7/1).
4/10	Four pushbuttons allow access for the specification of message options; 'Urgency', 'Notification', 'Date & Time', and 'Miscellaneous' (see IRM–5/1).
4/11	Message subject entered into a single-line editable text field.
4/12	Message body entered into a large scrolling multi-line text field.
4/13	Three buttons, on the bottom of the window, are available to Send, Cancel, or save the message into the drafts folder for later completion.

7.2.5 Message Options

The numerous message options available for the IPM class of UAs shall be classified into smaller and manageable related groups. Each shall be accessible from the new message composition window. The related groups are:

- urgency options
- notification options
- date & time options
- miscellaneous options

Urgency Options

The urgency options shall create a new window and display the importance and sensitivity options applicable to the message. The importance options are a series of radio buttons with possible values of:

- high
- normal
- low

The sensitivity options are a series of radio buttons with possible values of:

- none
- personal
- private
- confidential

The default value for the importance options should be set to 'normal'. The default value for the sensitivity options should be set to 'none'. The urgency options window shall also contain an 'ok' and 'cancel' button to manipulate the dialogue.

Notification Options

The notification options shall create a new window and provide access to setting the notification options applicable to the message. The notification options can be specified on a per recipient basis. Because of this constraint, if there are no nicknames in the message recipients list, then the notification options window shall not be displayed. Instead, an appropriate feedback message should be displayed. The notification options include:

- non-receipt notification
- receipt notification
- return message body
- reply requested

Each option should be displayed using a toggle button. By default, 'non-receipt notification' should be set to true, the rest to false. The IPM standard implies that the 'receipt notification' and 'return message body' may only be set if the 'non-receipt notification' is set to true. This should be reflected in the layout of the toggle buttons. Both the 'receipt notification' and 'return message body' toggle buttons should be indented under the 'non-receipt notification' toggle button and should be visibly disabled if the 'non-receipt notification' toggle button is set to false.

Since the notification options can be assigned to individual message recipients, an option menu shall be displayed for the selection of the recipient nickname. The list of nicknames shall be the currently selected list of message recipients. To provide a short-cut for the user, an extra nickname should be added to the option menu called 'All Recipients'. If this nickname is selected, then notification options shall apply to all message recipients.

The notification options window shall also contain an 'ok' and 'cancel' button to manipulate the dialogue.

Date & Time Options

The date & time options shall create a new window and display the time-dependent options applicable to the message. These options include:

- deferred delivery
- message expiration
- message reply by time

Each option shall be displayed as a series of toggle buttons with all being set to 'off' by default. Next to each option shall be a collection of option menus that can be manipulated to set the desired date and time. The default date should be set to some predetermined future date (possibly via some UA preferences setting or a fixed offset). The default time should be set to '12:00 AM'. If the toggle button is set to 'off', then all the date and time option menus shall be disabled.

Figure 114 shows an example of a possible implementation of part of the date & time options screen. The day option menu shall display all the available days consistent with the currently selected month. In the case that 24-hour time notation needs to be supported, the day option

menu shall display the numbers one to twenty-four and omit the AM/PM option menu. The minutes option menu shall display the sixty minutes in increments of five units.

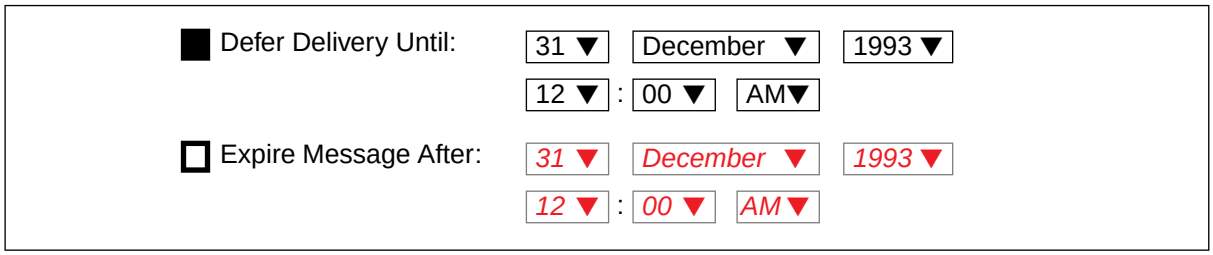


Figure 114: IRM—Date & Time Options Example

The date & time options window shall also contain an 'ok' and 'cancel' button to manipulate the dialogue.

Miscellaneous Options

The miscellaneous options shall create a new window and display various options applicable to the message. These options include:

- authoring user
- reply recipients
- obsoleted messages
- related messages

The first two options require O/R addresses (nicknames) and the latter two require IPM identifiers to be specified. The most effective method of specifying the 'authoring user' and 'reply recipients' shall be to provide a list of nicknames and allow the user to move the nickname(s) to corresponding lists representing the two options. This would operate in the same manner as message recipient specification described in section '7.2.4 Sending Messages'.

The entry of IPM identifiers for obsoleted and related messages can be a tedious and erroneous task as the identifiers are long and convoluted text strings. A productive alternative should be supplied on systems supporting a drag-n-drop interface. The user should be able to drag obsoleted and related messages from the message list screen into corresponding scrolling list fields on the miscellaneous options window. Alternatively, allow the user to specify the message by a system-defined numbering system.

Methods to remove the items in all four lists shall also be provided. The miscellaneous options window shall also contain an 'ok' and 'cancel' button to manipulate the dialogue.

Table 33 summarises the IRM message options recommendations.

Table 33: IRM—Message Options Recommendations

Number	IRM Recommendation
5/1	Message options should be grouped into four distinct groups; Urgency, Notification, Date & Time, and Miscellaneous.
5/2	Urgency options display the Importance and Sensitivity options as a series of radio buttons.
5/3	The values for Importance include; high, normal, and low. The default value is normal.

Table 33: IRM—Message Options Recommendations *(continued)*

Number	IRM Recommendation
5/4	The values for Sensitivity include; none, personal, private, and confidential. The default value is none
5/5	Notification options include; non-receipt, receipt, return message body, and reply requested options as a series of toggle buttons.
5/6	The default notification option is non-receipt set to on, all others are off.
5/7	The receipt and return message body option are dependent on the non-receipt option. Hence, they should be indented under this option and disabled if non-receipt is set to off.
5/8	Each notification option can be specified on a per recipient basis. An option menu should be available of the current list of message recipients. An extra item called 'All Recipients' shall also be available on this menu for setting the same options for all message recipients.
5/9	Date & Time options include; deferred delivery, message expiration, and message reply by times as a series of toggle buttons. The default values are off.
5/10	Displayed next to each Date & Time option is a collection of option menus for the entry of the date and time values. These include day, month, year, hour, minute, and am/pm option menus. If 24-hour notation is to be supported, the am/pm option menu may be omitted.
5/11	The Date & Time option menus are all disabled if the corresponding toggle buttons are off.
5/12	The default time is 12:00:AM and date should be a fixed interval into the future (possibly set by UA preferences).
5/13	Miscellaneous options include; authorising user, reply recipients, obsoleted messages, and related messages.
5/14	Authorising user and reply recipients require O/R addresses. A list of nicknames shall be used to select these options from (see IRM–4/3).
5/15	Obsoleted and related messages require IPM identifiers to be specified. A drag-n-drop interface should allow existing messages to be dragged to appropriate lists. Alternatively, allow the specification of the message via a system-defined numbering system.

7.2.6 Message Addressing

The key to the message addressing screen is to provide a mapping between the common forms of O/R representations and the screen display. The specification of O/R addresses shall be provided by an adaptable form-based screen. The window shall consist of a group of toggle buttons that each reflect one of the O/R address attributes. Each toggle button will display the attribute name and common acronym. The user is then able to select or de-select an attribute based on the source information they have. (For example, a user may have a business card showing an X.400 address with only five attributes listed.)

The bottom half of the screen shall contain text labels and fields to reflect the attribute/value relationship of the O/R address. Each attribute shall be displayed with the full attribute name and a text entry field. Preceding the text entry field will be the acronym that is common to that attribute with an equal sign between the two. Each attribute label and text field shall only be

displayed if the corresponding toggle button has been set to 'on'. If the attribute toggle button has been set to 'off' then the label and text field shall be unmapped from the window. The location of the text entry fields shall be consistent with the sequence of attribute toggle buttons.

Mandatory attributes shall always be displayed on the screen. These should include Country, ADMD, and PRMD. Hence, these attributes do not require attribute toggle buttons. By default, the mandatory, organisation, and surname attributes shall be displayed as these are the most common O/R address combinations. The Country attribute should be accessible via an option menu as the range of values is finite. Each item on the option menu shall display the full country name and common two character abbreviation. The country option menu shall default to the current country the user resides in.

Figure 115 shows an example implementation of the O/R addressing window. Since this screen size is dynamic, the window may grow to be larger than the user's display. In this case, the text entry fields shall be grouped together and displayed within a scrolling list.

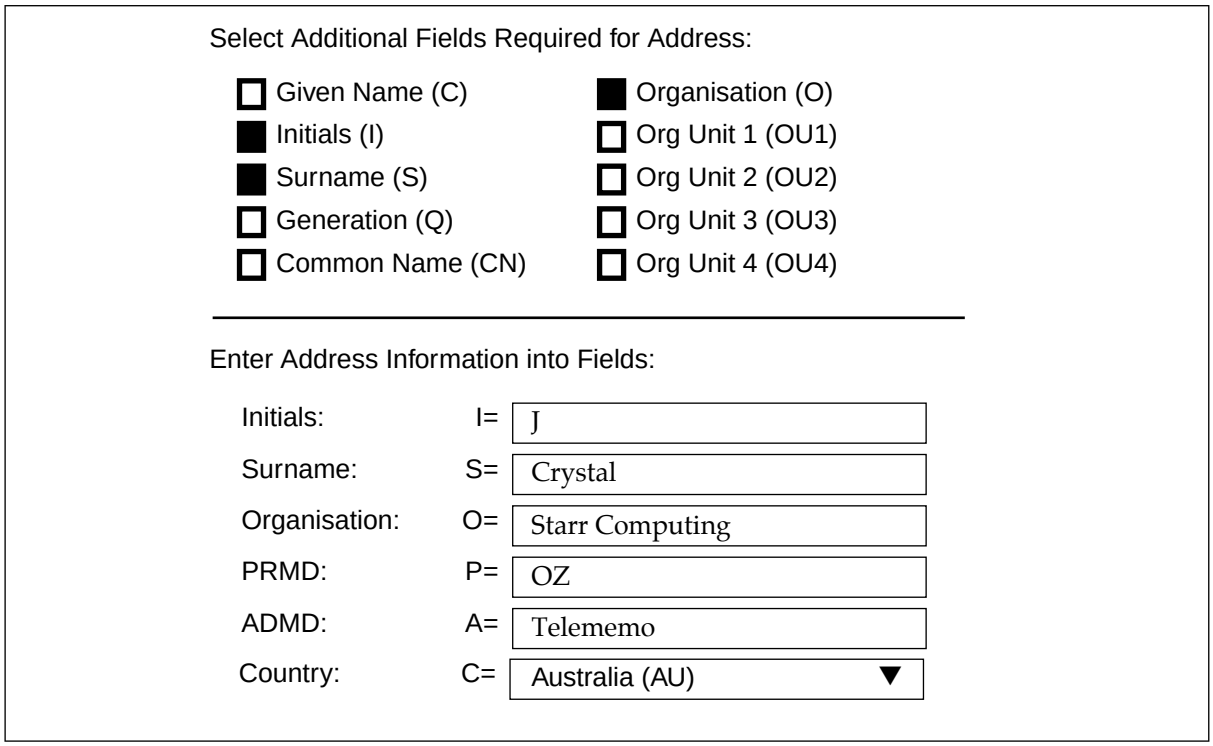


Figure 115: IRM—O/R Address Example

The O/R addressing window shall also contain an 'ok' and 'cancel' button to manipulate the dialogue. If 'ok' is selected, the user shall then enter a nickname to be associated with the O/R address.

Table 34 summarises the IRM message addressing recommendations.

Table 34: IRM—Message Addressing Recommendations

Number	IRM Recommendation
6/1	New O/R addresses shall be specified in an adaptable new window.
6/2	The O/R address attributes can be selected from a number of toggle buttons. Each toggle buttons relates to a single attribute name including the common acronym.

Table 34: IRM—Message Addressing Recommendations *(continued)*

Number	IRM Recommendation
6/3	The data is entered into a series of vertically aligned text entry fields. Each field relates to a single attribute and the acronym is displayed next to the text entry field with an equal sign in between.
6/4	The attribute toggle buttons control the display of each text entry field. If on, the field is displayed, if off, the field is removed from the screen.
6/5	The country attribute shall be an option menu of all valid country names and two-letter codes. The default value is the current country.
6/6	Mandatory attributes that shall always be displayed on the screen include Country, ADMD, and PRMD. No toggle buttons shall be required for these attributes.
6/7	Apart from the mandatory attributes, the other attributes displayed on the screen by default are the Surname and Organisation.
6/8	A nickname shall be associated with the entered O/R address and added to the nickname list.

7.2.7 Directory Integration

The integration of X.500 Directory services should enhance the user's ability to discover and retrieve recipient O/R addresses. The directory integration shall provide a new window with three attributes required from the user:

- name,
- organisation, and
- country.

Both the name and organisation shall be clearly labelled text entry fields and the country shall be an option menu of available countries. A 'search' and 'cancel' button shall also be provided on the directory screen.

The 'search' button shall activate the directory search algorithm to maximise the success of matching entries. The algorithm should attempt a large range of paths of the Directory Information Tree to provide a greater success rate. Such a quasi-intelligent algorithm has been presented in the Iris Directory Lookup Algorithm (see Appendix F).

During the search procedure, the process may require input from the user to select from a set of possible matched entries to narrow the scope of the search. This shall involve a new dialog screen with a scrolling list of possibilities to select from. The user shall also have the option to cancel the search at this stage. This process may involve a number of steps until a final list of matched entries is found. The user shall then select from this list and the recipient directory entry read to find the O/R address. If no O/R address attribute is found, then the user is informed and the search cancelled. If successful, the user shall be asked to enter a nickname to be associated with the O/R address.

Throughout all stages of the search process, appropriate feedback should be displayed with the option to cancel the search. Any error conditions (eg directory server unavailable) or failed matches should be reported.

Table 35 summarises the IRM directory integration recommendations.

Table 35: IRM—Directory Integration Recommendations

Number	IRM Recommendation
7/1	X.500 Directory Services shall be integrated with IPM UAs.
7/2	The directory search requires the recipients name, organisation, and country names. Name and organisation shall be entered into text fields, and country shall be selected via an option menu.
7/3	A comprehensive search 'engine' shall be utilised to maximise the success rate and searching paths of the directory.
7/4	Informative feedback as to the progress of the search process shall always be displayed.
7/5	Feedback from the user shall be obtained to assist in narrowing the search options in cases of multiple choices. A list of options should be presented to the user.
7/6	User shall be able to cancel the search process at any stage.
7/7	If search is successful, the recipients O/R address is read and automatically stored with an associated nickname supplied from the user.

7.2.8 User Agent Administration

The UA shall provide adequate facilities to manage the administration of IPM services. Such services include:

- deferred delivery message management
- automatic message forwarding
- automatic message acknowledgements
- automatic expired/obsolete message discarding

The deferred delivery management should provide a list of all messages currently queued for delivery at a future date. For each deferred message in the queue, the user should be able to:

- read and modify each message,
- remove the message,
- put the message on hold indefinitely, and
- send the message now.

All the other services can simply be provided with appropriate toggle buttons and text fields. The specification of the recipient for forwarding of messages shall be provided with an option menu of all current nicknames. Figure 116 shows an example of the IPM administration window.

A facility to manage the message folders and nicknames should also be provided by the UA. This should allow for the addition, modification, and removal of message folders and nicknames. The message folders management shall be provided with the ability to edit each folder group and individual folder names. This shall be provided on the main window near the message folders list. A scrolling list should be used, similar to the move message dialog window, to enable this facility. The nickname management shall be provided on the send message win-

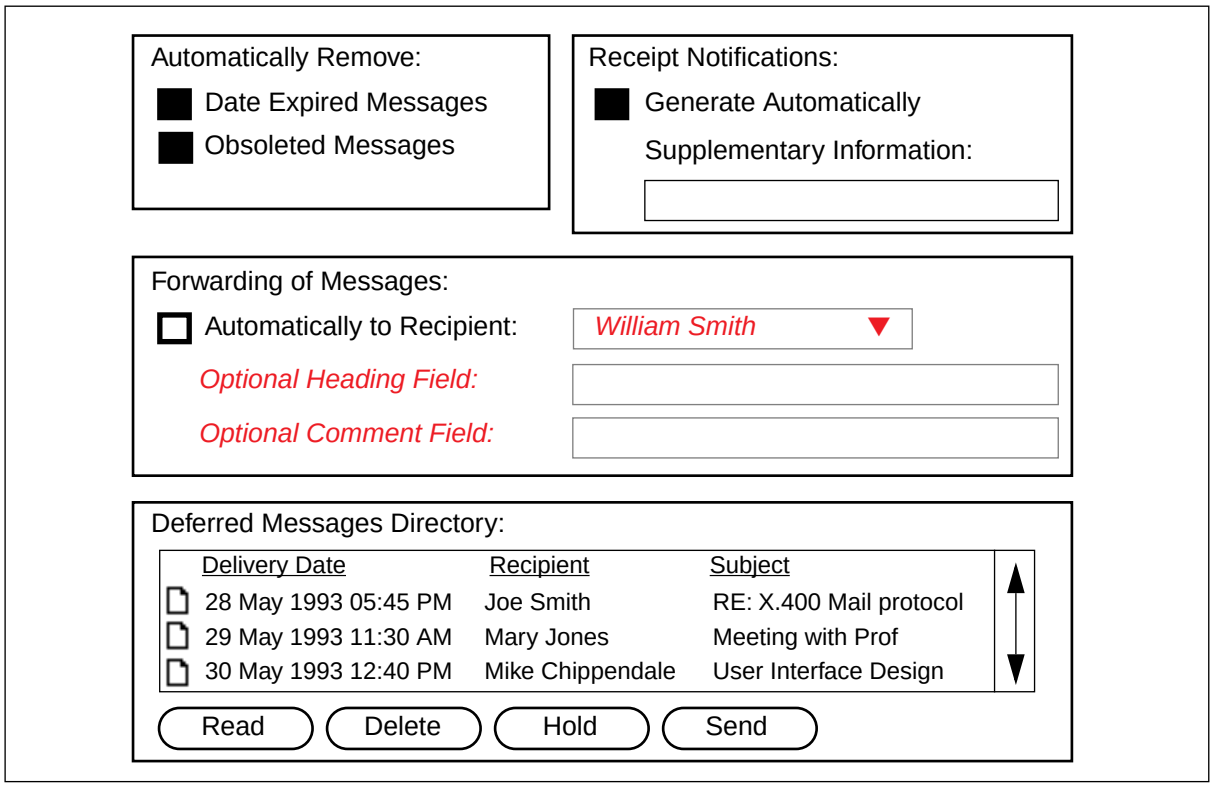


Figure 116: IRM—UA Administration Example

dow, directly under the nickname list. The standard O/R address screen shall be used to modify nickname entries.

Table 36 summarises the IRM UA administration recommendations.

Table 36: IRM—UA Administration Recommendations

Number	IRM Recommendation
8/1	A list of deferred delivery messages shall be available for review. The messages can be read and modified, deleted, held indefinitely, or sent immediately.
8/2	Messages can be automatically forwarded to another recipient. This toggle button is off by default. An option menu of nicknames is used to specify the recipient. Other optional information (heading and comment fields) are entered into text fields.
8/3	Message receipt notifications can be automatically generated. This toggle button should be on by default. Supplementary information is entered into a text field.
8/4	Two toggle buttons indicate if date-expired and obsoleted messages are to be automatically removed. The default is true.
8/5	Management of message folders and folder groups (creation, removal, and modification) shall be supported. A list of message folders can be used to edit the names.
8/6	Nickname management (creation, removal, and modification) shall be provided. The O/R address entry screen shall be used to modify existing nicknames.

7.3 Summary

The Iris Reference Model represents a comprehensive framework for the implementation of X.400 UAs on graphical user interface environments. The IRM designs have proven successful in implementations under OSF/Motif and should provide similar results on other software systems.

Finally, as an aid to the dissemination of X.400 addresses to closely match the IRM, it is strongly recommended that O/R address attributes be printed in a vertical format. That is, the attribute/value pairs are displayed in the form 'CN=Fred Smith' in a vertical layout. This would match the representation presented in the O/R address screen in the IRM.

Table 37 summarises the IRM O/R address layout recommendations.

Table 37: IRM—O/R Address Layout Recommendations

Number	IRM Recommendation
9/1	The printing of O/R addresses (eg on business cards) should follow a vertical layout. Each attribute/value pair should be displayed with the acronym and value.

Chapter 8
Discussion and Conclusion

8.1 Introduction

This chapter outlines the main conclusions of the research into MHS user agents with integrated directory support. Interpretations of the research findings and implications for researchers and practitioners, including limitations, are also discussed. Suggestions for some possible future work based on the research findings are also outlined. Finally, some comments on the future of messaging and society are examined.

8.2 Discussion of Research Findings

The research findings indicate that the original project objectives have been satisfied to the extent that conclusive findings have been reached. The original projects objectives for the UA reference model included research into:

- graphical user agent design for X.400 UAs
- X.400 message addressing
- X.500 integration to X.400 UAs
- large list management
- forms-based interface screens
- UIMS development
- user interface evaluation

The design of graphical user interfaces for X.400 UAs proved a non-trivial task. The functionality of the IPM standard was successfully transformed into an effective and novel graphical user interface for the Iris UA. The design of the graphical user interface for the Iris UA highlighted the need for such interfaces to support the wide range of IPM services. These include the use of innovative interaction techniques to reduce the load on the user.

The study of existing user interface guidelines focused attention on the user during the process of analysing the IPM UA functional requirements. The Iris UA design also benefited from the review of existing electronic mail user interfaces, thus providing some level of consistency

across the application domain. The design recognised the problem of the large number of IPM options and categorised these options into cohesive groups. Each group of options can now be handled with greater efficiency without overloading the user. The lack of support for drag-n-drop interaction with OSF/Motif did prove limiting in some areas. However, by utilising the existing user interface components and interaction techniques, similar functionality and utility was provided.

Message recipient specification is seen as a key area to provide an effective and natural process for UAs. Message addressing with X.400 O/R addresses was supported with the design of an adaptable forms-based screen. The mapping between the two representations—paper and screen—proved favourable which enabled effective input of large structured data. This requirement also provides a recommendation on the printing of O/R addresses to closely map the screen layout. The evaluations indicated that a vertical layout for O/R addresses provided greater success for data entry.

The integration of X.500 Directory Services into UAs proved to be a key ingredient to efficacious messaging. The importance of providing such a service is paramount to the success of the UA as it will be the preferred method for message addressing. The Iris UA used a simplistic yet opportune approach to provide this service.

The 'quasi-intelligence' of the Iris directory lookup algorithm empowered the user and provided a high level of search success. This transfer of interaction requirements to the search engine simulates a greater rapport with the user and, at times, invites the user to become part of the search process. The underlying use of X.500 Directory Services is as transparent to the user as the global database itself.

The development of the Iris UA with a commercial UIMS highlighted the immaturity of the software market for these tools at the time. The abandonment of the UIMS approach was disappointing as it was seen to be an effective methodology for integration into the software engineering lifecycle. The revised implementation plan proved to be just as effective and enabled closer control and granularity.

Some of the early work with the UIMS showed benefits in the speed of development and use of simulation techniques. However, development using the OSF/Motif Toolkit was just as effective after a level of experience was obtained. With appropriate libraries of common interface components, development of the Iris UA prototype at the toolkit level realised comparable results. The OSF/Motif widget set provided a sophisticated and extensive range of interface components that would complement any contemporary software interface.

The usability techniques used in the evaluation of the Iris UA emphasised the requirements for an effective and planned approach to user interface design. The success of the final interface was the direct result of the incorporation of the prototypical development lifecycle. The study of the evaluation results provided quick and direct feedback to the interface problem areas enabling subsequent interface changes.

The fundamental approach to the development of the task scenarios enabled realistic usability evaluation results. Each task was typical of daily use of electronic messaging systems and covered the majority of task applicable for users at an intermediate level. The video-taped sessions enabled a practical review process to be undertaken to identify user interface problems for redesign in subsequent stages. The follow-up questionnaires and task-completion times

were also effective in establishing quantitative measures for statistical analysis and substantiate the interface designs.

8.3 Thesis contributions

The major significant contribution presented in the research findings is the comprehensive Iris Reference Model (IRM). The Iris UA, which was the basis of the IRM, was proposed, designed, implemented, and finally, favourably evaluated. The successful implementation of the IRM, in the form of the Iris UA, proved conclusive and provides a benchmark for the judgement for other IPM UAs. The design of the IRM identified and analysed possible alternatives and presented a model based on the functional requirements of the IPM class of cooperating UAs. The idea that a graphical user interface reference model for application-layer protocols, such as X.400 and X.500, may also prove to be an important precursor to the development of many other HCI-centred standards.

The IRM contains novel interaction methods, particularly for message folder management, as well as providing a consistent interface to similar messaging applications. The IRM was based on an iterative design-implementation-evaluation process to provide justification for the recommendations. Parts of the IRM can be used as resources for non-X.400 messaging systems as well as other application domains. The IRM could conceivably be considered as an appendix to the X.400 MHS recommendation series as a guideline for implementors of IPM UAs.

A significant contribution to X.400 O/R addressing was presented in the form of an adaptable forms-based entry screen. The O/R address format has been heavily criticised for its complexity and machine-orientation. The provision of an adaptable screen for the entry of O/R addresses reduces the cognitive load on the user as direct mapping of the input data to the screen text fields can be achieved. The alternative, reproducing all possible text fields on the screen, will confuse the typical user.

The vertical attribute/value format of the entry screen has also produced a recommendation for the dissemination of O/R addresses printed on paper and other media to match the screen layout. The wider use of this user-adaptable screen format can easily be employed in other systems in which structured data entry is required.

The integration of X.500 Directory Services with IPM UAs has also been a significant contribution. The use of the directory to support message addressing is seen as probably the key to the successful deployment of X.400 MHS in an organisation. The provision of simple directory lookup facilities can easily be achieved by requiring a minimal amount of search information from the user. The fundamental aspect to the integration is the power of the directory searching framework. A comprehensive and exhaustive algorithm must be used to maximise the query results and provide seamless dialogue with the user. Such an implementation has been achieved in the Iris Directory Lookup Algorithm.

Another contribution is in the design and evaluation techniques used in the prototypical development process. The iterative design and evaluation steps, by both paper and computer-generated systems, proved successful. The design was based on sound principles and extensive citation of user interface guidelines. Adding heuristic evaluation proved to be a simple yet effective method. The major gains were evident in the use of user interface guidelines as a base for design decisions. The access to large sets of such guidelines in hypertext forms for browsing and gathering is an important requirement.

The scenario-based evaluation methodology, applied to the Iris interface, generated convincing results. The integration of a broad set of usability and performance evaluation techniques and the application of the results of the evaluations demonstrated the effectiveness of the methods. The design re-evaluation was a critical case study of a sound HCI design process. With the aid of video taped sessions, the user interface problems were effectively and accurately identified. The user satisfaction questionnaires and, in particular, the task-completion times also added strong evidence to the evaluation process. These usability techniques should be applicable for the evaluation of designs in other software system domains.

8.4 Future Work

The research findings have left some promising areas for future investigation. Most of these can be classified as implementation enhancements to enable the Iris UA to reach a professional and marketable level. Others are research areas that would require some form of usability evaluation. The IRM could then be updated to include such enhancements. A number of possible areas has been identified.

Comprehensive message management tools, such as searching and sorting, to provide a powerful database of messages. One of the problems with storing messages in folders is the possibility that they will seldom be accessed again. If facilities are implemented to enable the user to effectively interrogate the message folders, then the UA will provide added benefits to the user's work activities.

Implementation of the miscellaneous message options. These include UA administrative options, nickname, and message folder management. The provision of these services completes the functionality of the IPM UA. Message folder management should be implemented using a direct manipulation technique. That is, the group and folder names should be edited directly in the scrolling lists. One key feature for nickname management would be the ability for the system to search and update all O/R addresses that were obtained from the X.500 directory. The nicknames would then be current and would also identify if a nickname is no longer valid.

Expansion of the nickname scrolling list to include an alphabetical sorting capability. For large nickname lists, the user should be able to narrow the number of items down by indicating the first letter(s) of the required names. One implementation method would involve a number of small pushbuttons located on the left side of the nickname list. Each button would be a letter of the alphabet. Selecting one of the pushbuttons would then display only the nicknames starting with that letter.

Provide support for the complete set of O/R address forms by providing extra toggle button attributes. A number of other O/R address forms have been identified by the X.400 standard; Terminal, Numeric, and Postal. Although they probably will not be as common as the mnemonic form, they should be supported. The proposed new additional form for Internet style addresses could also be added to the O/R address entry screen.

Provide an iconic interface to the UA services similar to the personal computer based systems. Icons for each of the services of the Iris UA could be added to improve the appearance and to provide similarity to other common systems. A comprehensive usability evaluation for the possible icons would have to be undertaken.

Support for directory browsing for experienced users. The X.500 directory provides opportunity to 'discover' the addresses of potential recipients. This would involve a 'browsing' inter-

face be supplied to enable the user to search any arbitrary organisation. Some work has already been completed in this area.

Support for the MIME messaging standard. Since MIME is gaining wider acceptance, the mapping of the Iris UA to a MIME UA would be a sensible expansion. The popularity of MIME and the potential to supply matching services over existing message transfer systems, cannot be overlooked. The requirements for MIME-based UAs are similar to IPM UAs and both could interoperate successfully.

As can be clearly seen, the future work is substantial which is typical for implementations of such complex and rich international standards. The key would be to keep the IRM consistent with any changes and enhancements made and provide evidence to the successful deployment of new services.

8.5 Messaging and Society

The importance of messaging in the future is both promising and empowering. Messaging services will play a larger and more significant role in society as the underlying technologies reach the public user in their homes. The implications of which, apart from the utility of instant communications, will require graphical user interfaces that are tightly integrated into home-based services. These may include a system to order the weekly groceries that, in fact, sends an electronic message to the local supermarket. The user, however, is totally unaware of this message transfer service.

Regardless of the future for the X.400 or MIME standards, it is clear that the user interfaces to messaging systems will become a prominent key technology. The services provided by X.500 directories will enhance the daily communications needs for a wide range of users. The advantages of electronic mail with integrated directory functionality for research, business, and society can only be strengthened with effective and productive graphical user interfaces.

Appendices

Appendix A: X.400 Message Transfer Service Elements

Basic Elements of Service
Access management
Content type indication
Converted indication
Delivery time stamp indication
Message indication
Non-delivery notification
Original encoded information types indication
Submission time stamp indication
User/UA capabilities registration

.

Optional Elements of Service	Essential or Additional
Alternate recipient allowed	E
Alternate recipient assignment	A
Content confidentiality	A
Content integrity	A
Conversion prohibition	E

Optional Elements of Service	Essential or Additional
Conversion prohibition in case of loss of information	A
Deferred delivery	E
Deferred delivery cancellation	E
Delivery notification	E
Designation of recipient by directory name	A
Disclosure of other recipients	E
DL expansion history indication	E
DL expansion prohibited	A
Explicit conversion	A
Grade of delivery selection	E
Hold for delivery	A
Implicit conversion	A
Latest delivery designation	A
Message flow confidentiality	A
Message origin authentication	A
Message security labelling	A
Message sequence integrity	A
Multi-destination delivery	A
Non-repudiation of delivery	A
Non-repudiation of origin	A
Non-repudiation of submission	A
Originator requested alternate recipient	A
Prevention of non-delivery notification	A
Probe	E
Probe origin authentication	A
Proof of delivery	A
Proof of submission	A
Redirection disallowed by originator	A
Redirection of incoming messages	A
Report origin authentication	A
Requested delivery method	E
Restricted delivery	A
Return of content	A

Optional Elements of Service	Essential or Additional
Secure access management	A
Use of distribution list	A

Appendix B: X.400 Physical Delivery Service Elements

Basic Elements of Service
Basic physical rendition
Ordinary mail
Physical forwarding allowed
Undeliverable mail with return of physical message

Optional Elements of Service	Essential or Additional
Additional physical rendition	A
Counter collection	E
Counter collection with advice	A
Delivery via Bureaufax service	A
EMS (express mail service)	E
Physical delivery notification by MHS	A
Physical delivery notification by PDS	A
Notification forwarding prohibited	A
Registered mail	A
Registered mail to addressee in person	A
Request for forwarding address	A
Special delivery	E

Appendix C: X.400 Message Store Service Elements

Basic Elements of Service
Stored message deletion
Stored message fetching
Stored message listing
Stored message summary

Optional Elements of Service	Essential or Additional
Stored message alert	A
Stored message auto-forward	A

Appendix D: X.400 IPM Service Elements

Optional Elements of Service	Essential or Additional	
	Origination	Reception
Additional physical rendition	A	A
Alternate recipient allowed	A	A
Authorizing users indication	A	E
Auto-forwarded indication	A	E
Basic physical rendition	A	E
Blind copy recipient indication	A	E
Body part encryption indication	A	E
Content confidentiality	A	A
Content integrity	A	A
Conversion prohibition	E	E
Conversion prohibition in case of loss of information	A	N/A
Counter collection	A	E
Counter collection with advice	A	A
Cross-referencing indication	A	E
Deferred delivery	E	N/A
Deferred delivery cancellation	A	N/A
Delivery notification	E	N/A
Delivery via Bureaufax service	A	A
Designation of recipient by directory name	A	N/A
Disclosure of other recipients	A	E
DL expansion history indication	N/A	E
DL expansion prohibited	A	A
EMS (express mail service)	A	E
Expiry date indication	A	E
Explicit conversion	A	N/A
Forwarded IP-message indication	A	E
Grade of delivery selection	E	E
Importance indication	A	E
Incomplete copy indication	A	A
Language indication	A	E

Optional Elements of Service	Essential or Additional	
	Origination	Reception
Latest delivery designation	A	N/A
Message flow confidentiality	A	N/A
Message origin authentication	A	A
Message security labelling	A	A
Message sequence integrity	A	A
Multi-destination delivery	E	N/A
Multi-part body	A	E
Non-receipt notification request indication	A	E
Non-repudiation of delivery	A	A
Non-repudiation of origin	A	A
Non-repudiation of submission	A	A
Obsoleting indication	A	E
Ordinary mail	A	E
Originator indication	E	E
Originator requested alternate recipient	A	N/A
Physical delivery notification by MHS	A	A
Physical delivery notification by PDS	A	E
Physical forwarding allowed	A	E

Optional Elements of Service (for a period of time)	Essential or Additional
Alternate recipient assignment	A
Hold for delivery	A
Implicit conversion	A
Redirection of incoming messages	A
Restricted delivery	A
Secure access management	A
Stored message alert	A
Stored message auto-forward	A

Appendix E: Smith & Mosier Guideline References

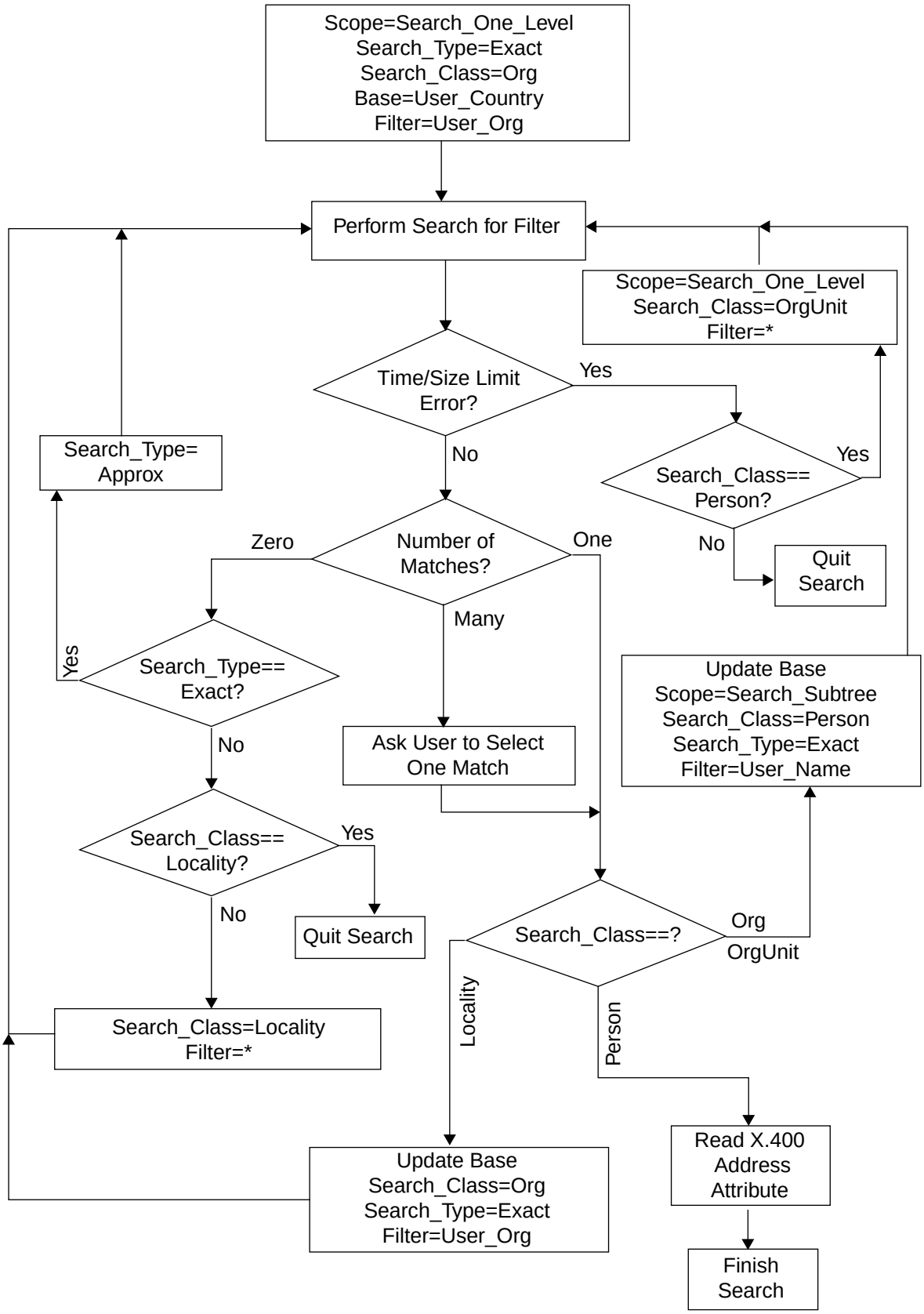
1.4/5	Data Field Labels	For each data field, display an associated label to help users understand what entries can be made.
1.4/18	Label Punctuation as Entry Cue	The label for each entry field should end with a special symbol, signifying that an entry may be made.
1.4/19	Informative Labels	In labeling data fields, employ descriptive wording, or else standard, predefined terms, codes and/or abbreviations; avoid arbitrary codes.
1.4/24	Form Compatible for Data Entry and Display	When forms are used for reviewing displayed data as well as for data entry, make the form for data entry compatible with that for display output; use the same item labels and ordering for both.
1.4/25	Form Compatible with Source Documents	When data entry involves transcription from source documents, ensure that form-filling displays match (or are compatible with) those documents, in terms of item ordering, data grouping, etc.
1.8/1	Default Values	When likely default values can be defined for the data entries in a particular task, offer those default values to speed data entry.
2.0/6	Consistent Display Format	For any particular type of data display, maintain consistent format from one display to another.
2.1/25	Hierarchic Structure for Long Lists	For a long list, extending more than one displayed page, consider adopting a hierarchic structure to permit its logical partitioning into related shorter lists.
3.1.2/1	Form Filling for Data Entry	Consider form filling for tasks where some flexibility in data entry is needed, such as the inclusion of optional as well as required items, where users will have moderate training, and/or where computer response may be slow.
3.1.3/20	Menus Distinct from Other Displayed Information	If menu options are included in a display that is intended also for data review and/or data entry, which is often a practical design approach, ensure that they are distinct from other displayed information; locate menu options consistently in the display and incorporate some consistent distinguishing feature to indicate their special function.
4.2/1	Consistent Feedback	Ensure that every input by a user will consistently produce some perceptible response output from the computer; when a terminal is in use, its display screen should never be blank.
5.0/2	Functional Wording	Choose functional wording for the terms used in data transmission—for preparing and addressing messages, for initiating and controlling message transmission and other forms of data transfer, and for receiving messages—so that those terms will match users' work-oriented terminology.

5.0/3	Consistent Procedures	Ensure that procedures for preparing, sending and receiving messages, are consistent from one transaction to another, and are consistent with procedures for other information handling tasks.
5.0/4	Minimal Memory Load on User	Design the data transmission procedures to minimize memory load on the user.
5.0/5	Minimal User Actions	Design the data transmission procedures to minimize required user actions.
5.0/6	Control by Explicit User Action	Design the data transmission procedures so that both sending and receiving messages are accomplished by explicit user action.
5.0/7	Flexible User Control	Provide for flexible control of data transmission, so that users can decide what data should be transmitted, when, and where.
5.0/9	Flexible Message Filing	In applications requiring general-purpose message handling, provide users with flexible capabilities for filing copies of draft messages during preparation, transmitted messages, and received messages, and organizing those message files.
5.0/10	Message Highlighting	Provide software capabilities to annotate transmitted data with appropriate highlighting to emphasize alarm/alert conditions, priority indicators, or other significant second-order information that could affect message handling.
5.1/1	Message Composition Compatible with Data Entry	Ensure that procedures for composing messages are compatible with general data entry procedures, especially those for text editing.
5.1/3	Unformatted Text	Allow users to compose and transmit messages as unformatted text.
5.1/7	Data Forms	In preparing data forms for transmission, allow users to enter, review, and change data on an organized display with field labels, rather than requiring users to deal with an unlabeled string of items.
5.1/9	Flexible Data Specification	Provide users with flexible means for specifying the data to be transmitted.
5.1/12	Variable Message Length	Allow users to prepare messages of any length.
5.1/13	Saving Draft Messages	Allow users to save draft messages during their preparation, or upon their completion.
5.2/1	Destination Selection	Allow users to specify the destination(s) to which data will be transmitted.
5.2/2	Standard Address Header	For addressing and identifying messages, provide a basic set of header fields that can be interpreted by all systems to which users will send messages.
5.2/3	Prompting Address Entry	When a user must specify the address for a message, provide prompting to guide the user in that process.

5.2/4	Address Directory	Provide users with a directory showing all acceptable forms of message addressing for each destination in the system, and for links to external systems.
5.2/5	Aids for Directory Search	Provide computer aids so that a user can search an address directory by specifying a complete or partial name.
5.2/6	Extracting Directory Addresses	Allow users to extract selected addresses from a directory for direct insertion into a header in order to specify the destination(s) for a message.
5.2/7	User-Assigned Nicknames for Addressing	Allow users to define nicknames for formal addresses, to save those nicknames in their own files, and to specify those nicknames when addressing messages.
5.2/15	Addressing Replies to Messages Received	If a user wishes to reply to a received message, provide the appropriate address(es) automatically, along with a reference to that received message.
5.2/16	Editing Address Headers	Allow users to edit the address fields in the header of a message being prepared for transmission.
5.2/17	Single Occurrence of Address	Ensure that the address of any particular recipient will occur only once in a message.
5.2/19	Redistributing Received Messages	Allow users to redistribute received messages by enlarging their address headers.
5.3/2	User Review Before Transmission	When computer aids are provided for preparing, addressing, and initiating message transmission, allow users to review and change messages prior to transmission.
5.3/7	Assignment of Priority	When messages will have different degrees of urgency, i.e., different implications for action by their recipients, allow the sender of a message to designate its relative priority, or else have the computer assign priority automatically.
5.3/9	Deferring Message Transmission	Allow users to defer the transmission of prepared messages, to be released by later action.
5.3/10	Transmission at Specified Date/Time	Allow users to specify a date and time for transmitting prepared messages.
5.3/11	Return Receipt	Provide for message transmission with 'return receipt requested' if users will require that capability.
5.3/12	Cancel Transmission	Allow senders to cancel a request for transmission of a message that has not yet been sent.
5.3/13	Printing Messages	Allow users to print copies of transmitted messages in order to make hard-copy records.
5.4/2	Automatic Feedback	Provide automatic feedback for data transmission confirming that messages have been sent or indicating transmission failures, as necessary to permit effective user participation in message handling.

5.4/3	User Specification of Feedback	Allow users to specify what feedback should be provided for routine message transmission, and also to request specific feedback for particular messages.
5.4/7	Notification of Transmission Failure	If message transmission is not successful, notify the sender and if possible include an explanation of the problem.
5.5/3	Queuing Messages Received	Provide default logic so that, unless otherwise specified by a user, the computer will route incoming messages automatically to a queue ('in-box'), pending subsequent review and disposition by the user.
5.5/5	Nondisruptive Notification of Arriving Messages	For messages arriving while a user is logged on to a system, ensure that notification of message arrival will not interfere with that user's other information handling tasks.
5.5/6	Indicating Priority of Received Messages	In applications where incoming messages will have different degrees of urgency, i.e., different implications for action, notify recipients of message priority and/or other pertinent information.
5.5/9	Information about Queued Messages	Allow users to review summary information about the type, source, and priority of queued incoming messages.
5.5/11	Specifying Format for Message Listings	Provide some means for users to specify the order in which header fields are displayed in messages and in message summary listings.
5.5/12	User Review of Messages in Queue	Provide convenient means for user review of received messages in their incoming queue, without necessarily requiring any further disposition action, i.e., without removal from the queue.
5.5/14	Message Review Compatible with Data Display	Ensure that computer aids and procedures for reviewing messages are consistent with other system capabilities for general data display.
5.5/18	Discarding Messages	Allow users to discard unwanted messages without filing them, or even without reading them in applications where 'junk mail' may be received.

Appendix F: Iris Directory Lookup Algorithm



Appendix G: Usability Consent Form

Our group is studying how people use computers. This will help us understand how to make computer software more useful to people. To do this, we need the help of people like you, so that we can observe people while they do particular tasks with computers. We are asking you to help us.

We will ask you to do one or more tasks, to be explained shortly, that will involve your use of some computer software. The observations that we make will be of considerable importance. The task itself might be very easy or very difficult. It might be quite interesting or it might be uninteresting. Before we start, we will tell you as much as we can about the task and our procedures, but we won't give you information that might affect how you do the task. After everything is finished, we will be happy to answer any other questions you have.

We want you to understand why we need your help. We want your participation to be a cooperative venture with us in adding to our understanding of how to make computers easier to use. In addition, we are required to get your written consent to participate. If you have any questions about this document, please ask us.

1. Task Description: To evaluate the usability of an Electronic Messaging System. You will be video-taped during your use of the computer so that we may review the session later. You should try to work as efficiently as possible but do not worry if you cannot complete a task. You will be asked some questions about your preferences after completing the tasks. The scenarios page of this form will give you more information about the task we will ask you to do.
2. No Risks: There are no risks involved in this task.
3. Questions: We will answer any questions you have about the task, except information that might affect your performance, which may be withheld until we are done. We will answer any question you have then. After the experiment is over, we would like you to not discuss the experiment with others who might participate because it could affect their performance.
4. Voluntary Participation: You may leave an any time before or during your task if you feel uncomfortable. We want you to continue because you want to cooperate, not because you feel obliged. If you are being paid for your participation you will get paid proportional to the time you have spent.
5. Confidentiality: Any public presentations or published reports of the results using your data will not include any means of identifying you, in order to provide confidentiality and privacy. We would be happy to provide to you any published reports on this work.
6. Evaluation of Software, Not Users: Finally, it is not you who is being evaluated in this study; it is the computer software that is being evaluated, so there are no right or wrong ways for you to perform.

Please sign on the line below if you agree to help us with this research.

Name: _____

Signature: _____

Date: _____

Appendix H: User Experience Questionnaire Results

Usability Trial One

Question	Trialist Number					
	1	2	3	4	5	6
Experience with computers	<3	>3	<3	<3	<1	>3
Experience with text-based interface electronic mail systems	<2	<1	0	<2	<1	<1
Experience with graphical interface electronic mail systems	<2	<2	<1	<1	<1	<2
Experience with X.400 electronic mail systems	0	0	0	0	0	0

Usability Trial Two

Question	Trialist Number					
	1	2	3	4	5	6
Experience with computers	>3	>3	>3	>3	>3	>3
Experience with text-based interface electronic mail systems	>3	<2	<2	<1	<1	<1
Experience with graphical interface electronic mail systems	<1	0	<1	<1	<1	<1
Experience with X.400 electronic mail systems	0	0	0	0	0	0

Note:

All results are in number of years.

Appendix I: Functionality Questionnaire Results—Trial One

Question	Trialist Number					
	1	2	3	4	5	6
read a mail message	5	4	4	4	5	4
reply to a mail message	5	5	4	4	4	4
send a mail message	1	3	3	4	4	2
move a mail message to a folder	5	3	4	4	3	4
move to different mail folders	5	4	4	2	4	4
delete mail messages	5	4	4	4	5	4
print mail messages	5	5	4	3	5	4
message options	2	3	5	2	2	5
recipient options	2	2	3	3	2	2
using the nicknames	3	2	4	4	1	5
given an X.400 address	1	2	4	2	1	3
using directory searching	4	2	3	4	3	4
adequate functions for electronic mail	4	5	4	4	5	5
easy access to all options for electronic mail	3	4	4	3	3	4
support for appropriate electronic mail tasks	4	5	4	3	3	4

Note:

1 = Strongly Disagree
2 = Disagree
3 = Neutral
4 = Agree
5 = Strongly Agree

Appendix J: Functionality Questionnaire Results—Trial Two

Question	Trialist Number					
	1	2	3	4	5	6
read a mail message	5	5	5	4	4	4
reply to a mail message	5	5	5	4	4	5
send a mail message	5	5	3	5	4	3
move a mail message to a folder	5	5	4	4	5	4
move to different mail folders	5	5	3	3	2	4
delete mail messages	5	5	5	5	5	5
print mail messages	5	5	5	5	5	5
message options	4	4	3	3	4	4
recipient options	4	4	4	4	4	4
using the nicknames	5	5	4	4	4	3
given an X.400 address	4	5	3	4	2	4
using directory searching	5	4	4	4	4	4
adequate functions for electronic mail	5	5	5	5	5	4
easy access to all options for electronic mail	4	5	4	5	5	4
support for appropriate electronic mail tasks	4	5	5	5	5	5

Note:

1 = Strongly Disagree
2 = Disagree
3 = Neutral
4 = Agree
5 = Strongly Agree

Appendix K: Interface Questionnaire Results—Trial One

Question	Trialist Number					
	1	2	3	4	5	6
Does the information appear to be organised logically on the screen	4	4	4	4	3	4
It is easy to find the required information on the screen	4	4	3	3	2	4
Is the information (eg menus, buttons, lists) displayed in a consistent location and layout	5	3	4	5	5	5
Is the way the system respond to a user actions consistent at all times	4	3	4	3	4	5
Is the format of displayed information compatible with the users perception	4	3	3	3	1	5
Does the sequence of activities required to complete a task follow what the users would expect	5	4	4	3	4	2
Are messages and instructions relevant and clear to the user	4	4	4	3	3	4
Are status messages informative and accurate	2	3	4	3	3	5
Where the user is presented with a list of options, is it clear as to what each option means	4	4	3	2	4	4
Is it clear what different parts of the system do	5	3	4	2	2	4
Do users have control over the order of information entry	5	3	4	4	5	3
Is the system flexible in allowing the user to choose options	5	3	4	4	5	5
Does the system allow the user to easily correct data	5	3	4	4	1	2
Does the system protect against errors in user actions	5	3	4	4	1	2
Overall, the interface was pleasing and easy to use	4	4	4	3	3	4

Note:

1 = Never
2 = Some of the Time
3 = Neutral
4 = Most of the Time
5= Always

Appendix L: Interface Questionnaire Results—Trial Two

Question	Trialist Number					
	1	2	3	4	5	6
Does the information appear to be organised logically on the screen	4	5	4	4	4	4
It is easy to find the required information on the screen	4	4	4	4	2	4
Is the information (eg menus, buttons, lists) displayed in a consistent location and layout	5	5	3	5	2	5
Is the way the system respond to a user actions consistent at all times	5	5	4	5	4	3
Is the format of displayed information compatible with the users perception	4	4	4	5	4	4
Does the sequence of activities required to complete a task follow what the users would expect	4	4	5	5	4	4
Are messages and instructions relevant and clear to the user	4	4	4	5	4	4
Are status messages informative and accurate	5	4	4	5	4	5
Where the user is presented with a list of options, is it clear as to what each option means	3	5	3	5	4	4
Is it clear what different parts of the system do	4	5	4	4	4	3
Do users have control over the order of information entry	5	4	5	5	4	4
Is the system flexible in allowing the user to choose options	5	5	5	5	4	4
Does the system allow the user to easily correct data	4	5	2	5	4	3
Does the system protect against errors in user actions	5	4	3	5	3	3
Overall, the interface was pleasing and easy to use	4	5	4	4	4	4

Note:

1 = Never
2 = Some of the Time
3 = Neutral
4 = Most of the Time
5= Always

Appendix M: Scenario Task Time Summary

Usability Trial One

Task Number	Trialist Number						Average
	1	2	3	4	5	6	
1	222	115	385	340	216	210	248
2	170	97	138	107	128	235	145
3	275	348	755	455	321	358	418
4	310	245	425	594	520	392	414
5	270	318	350	377	510	230	342

Usability Trial Two

Task Number	Trialist Number						Average
	1	2	3	4	5	6	
1	180	87	102	125	125	165	130
2	52	84	45	75	98	170	87
3	345	255	265	232	275	475	307
4	247	330	283	305	178	380	287
5	285	225	210	200	210	225	225

Note:

All results are in seconds

Appendix N: Statistical Test Results

Note:

FQ–T1 = Functionality Questionnaire—Trial One
FQ–T2 = Functionality Questionnaire—Trial Two
IQ–T1 = Interface Questionnaire—Trial One
IQ–T2 = Interface Questionnaire—Trial Two
TT–T1 = Task Times—Trial One
TT–T2 = Task Times—Trial Two

TEST	N	MEAN	MEDIAN	TRMEAN	STDEV	SEMEAN
FQ–T1	15	3.587	3.830	3.626	0.746	0.193
FQ–T2	15	4.367	4.500	4.369	0.455	0.117
IQ–T1	15	3.633	3.670	3.602	0.413	0.107
IQ–T2	15	4.1733	4.2000	4.1692	0.2789	0.0720
TT–T1	5	313.9	342.5	313.9	116.7	52.2
TT–T2	5	207.8	225.8	207.8	96.3	43.1

TEST	MIN	MAX	Q1	Q3
FQ–T1	2.170	4.500	3.170	4.330
FQ–T2	3.700	5.000	4.000	4.800
IQ–T1	3.170	4.500	3.330	3.830
IQ–T2	3.7000	4.7000	4.0000	4.3000
TT–T1	145.8	418.7	196.9	416.5
TT–T2	87.3	307.8	109.0	297.5

FQ–T1 Versus FQ–T2

N	MEAN	STDEV	SE MEAN	T	P VALUE
15	0.779	0.504	0.130	5.99	0.0000

N	N FOR TEST	WILCOXON STATISTIC	P-VALUE	ESTIMATED MEDIAN
15	15	119.0	0.000	0.7700

IQ–T1 Versus IQ–T2

N	MEAN	STDEV	SE MEAN	T	P VALUE
15	0.5400	0.3250	0.0839	6.44	0.0000

N	N FOR TEST	WILCOXON STATISTIC	P-VALUE	ESTIMATED MEDIAN
15	15	119.0	0.000	0.5250

TT–T1 Versus TT–T2

N	MEAN	STDEV	SE MEAN	T	P VALUE
5	-106.1	27.2	12.2	-8.71	0.0005

N	N FOR TEST	WILCOXON STATISTIC	P-VALUE	ESTIMATED MEDIAN
5	5	0.0	0.030	-114.1

Appendix O: IRM Recommendations

Number	IRM Recommendation
Message List Recommendations	
1/1	Message list displayed in prominent section of the main window
1/2	One-line summary of each message (date, originator, subject) displayed in the message list. Message urgency should also be displayed. Date format determined by the default operating system. Originator's name to be displayed rather than the O/R address.
1/3	Read, reply, forward, move, print, and delete functions to be supported for each message and displayed under the message list. Default function is to read a message (see IRM–3/1).
1/4	Reply function generates a new message (see IRM–4/1). The message is pre-addressed to the originator and any reply-recipients. Message subject preceded with 'RE: '.
1/5	Forward function generates a new message (see IRM–4/1). Body of original message text is included in the forwarded message surrounded by indicators. Message subject preceded with 'FWD: '.
1/6	Print and delete functions confirm their actions.
1/7	Move option displays a list of all message folders (a one-dimensional representation) to move message to. If drag-n-drop environment supported, then message icon can be dragged over message folder icon (see IRM–2/5).
1/8	Current message folder name displayed above the message list.
1/9	The default folder name for new messages is 'inbox' (see IRM–2/7).
1/10	Message list can be sorted by date, originator, subject, or priority.
1/11	Message list should display a minimum of five messages.
Message Folder Recommendations	
2/1	Message folder management facilities (add, remove, modify) should be provided
2/2	Message folders are associated into distinct groups. Each group being a list of message folders.
2/3	A two-dimensional view of the message folders shall be displayed on the main window in scrolling lists. Each group name is displayed above a scrolling list of the associated folder names.
2/4	Double-clicking on the message folder name will change the list displayed in the message list (see IRM–1/8).
2/5	If drag-n-drop can be supported, then each message folder name is displayed with a generic folder icon. This enables a message to be dragged over the icon for filing.
2/6	A dialog box interface for moving messages to the folders shall also be supported. The dialog will contain a single list of all message folders, indented under their message group names, for user selection.

Number	IRM Recommendation
2/7	A default 'System' message group shall be available. This shall contain the 'inbox' folder (for new messages) and other system-dependent folders (eg drafts and sent-mail).
2/8	The 'sent-mail' folder should automatically, at end of each month, be either removed or saved into another date-specific folder.
	Reading Messages Recommendations
3/1	Reading a message displays a new window with the IPM details shown.
3/2	The message body shall be displayed in a large scrolling text field.
3/3	The key heading fields (originator, subject, date) shall be displayed in separate text fields.
3/4	Other message heading fields should be displayed in a single scrolling field in an 'attribute=value' format.
3/5	Message operations, including reply, forward, move, print, and delete, shall be provided on the message window.
	Sending Messages Recommendations
4/1	New messages shall create a new window for the entry of the IPM data.
4/2	New messages shall be created from a 'Compose New Message' button on the main window.
4/3	A scrolling list of nicknames shall be available for the specification of message recipients. The message recipients shall be displayed in a separate Recipient list.
4/4	Nicknames can be moved from/to the recipient list via double-clicking on the name.
4/5	Two buttons; 'Add >>' and '<< Remove' shall also allow nicknames to be added/removed from the recipient list with single selection of the nickname. The buttons are located between the nickname and recipient lists.
4/6	Systems supporting drag-n-drop shall allow nicknames to be dragged from/to the nickname and recipients lists. In this case, a small generic icon will be displayed adjacent to each nickname.
4/7	Three recipient categories shall be supported; primary, carbon-copy, and blind-copy. These shall be displayed as a series of toggle buttons between the nickname and recipients lists.
4/8	Additions to the recipient list will display the category specification selected via the toggle buttons. These shall be 'To: ', 'Copy: ', or 'Blind: '.
4/9	Two buttons allow the creation of new nicknames recipients; 'Enter X.400 Address' (see IRM–6/1) and 'Lookup Name in Global Directory' (see IRM–7/1).
4/10	Four pushbuttons allow access for the specification of message options; 'Urgency', 'Notification', 'Date & Time', and 'Miscellaneous' (see IRM–5/1).
4/11	Message subject entered into a single-line editable text field.
4/12	Message body entered into a large scrolling multi-line text field.
4/13	Three buttons, on the bottom of the window, are available to Send, Cancel, or save the message into the drafts folder for later completion.

Number	IRM Recommendation
Message Options Recommendations	
5/1	Message options should be grouped into four distinct groups; Urgency, Notification, Date & Time, and Miscellaneous.
5/2	Urgency options display the Importance and Sensitivity options as a series of radio buttons.
5/3	The values for Importance include; high, normal, and low. The default value is normal.
5/4	The values for Sensitivity include; none, personal, private, and confidential. The default value is none
5/5	Notification options include; non-receipt, receipt, return message body, and reply requested options as a series of toggle buttons.
5/6	The default notification option is non-receipt set to on, all others are off.
5/7	The receipt and return message body option are dependent on the non-receipt option. Hence, they should be indented under this option and disabled if non-receipt is set to off.
5/8	Each notification option can be specified on a per recipient basis. An option menu should be available of the current list of message recipients. An extra item called 'All Recipients' shall also be available on this menu for setting the same options for all message recipients.
5/9	Date & Time options include; deferred delivery, message expiration, and message reply by times as a series of toggle buttons. The default values are off.
5/10	Displayed next to each Date & Time option is a collection of option menus for the entry of the date and time values. These include day, month, year, hour, minute, and am/pm option menus. If 24-hour notation is to be supported, the am/pm option menu may be omitted.
5/11	The Date & Time option menus are all disabled if the corresponding toggle buttons are off.
5/12	The default time is 12:00:AM and date should be a fixed interval into the future (possibly set by UA preferences).
5/13	Miscellaneous options include; authorising user, reply recipients, obsoleted messages, and related messages.
5/14	Authorising user and reply recipients require O/R addresses. A list of nicknames shall be used to select these options from (see IRM–4/3).
5/15	Obsoleted and related messages require IPM identifiers to be specified. A drag-n-drop interface should allow existing messages to be dragged to appropriate lists. Alternatively, allow the specification of the message via a system-defined numbering system.
Message Addressing Recommendations	
6/1	New O/R addresses shall be specified in an adaptable new window.
6/2	The O/R address attributes can be selected from a number of toggle buttons. Each toggle buttons relates to a single attribute name including the common acronym.

Number	IRM Recommendation
6/3	The data is entered into a series of vertically aligned text entry fields. Each field relates to a single attribute and the acronym is displayed next to the text entry field with an equal sign in between.
6/4	The attribute toggle buttons control the display of each text entry field. If on, the field is displayed, if off, the field is removed from the screen.
6/5	The country attribute shall be an option menu of all valid country names and two-letter codes. The default value is the current country.
6/6	Mandatory attributes that shall always be displayed on the screen include Country, ADMD, and PRMD. No toggle buttons shall be required for these attributes.
6/7	Apart from the mandatory attributes, the other attributes displayed on the screen by default are the Surname and Organisation.
6/8	A nickname shall be associated with the entered O/R address and added to the nickname list.
Directory Integration Recommendations	
7/1	X.500 Directory Services shall be integrated with IPM UAs.
7/2	The directory search requires the recipients name, organisation, and country names. Name and organisation shall be entered into text fields, and country shall be selected via an option menu.
7/3	A comprehensive search 'engine' shall be utilised to maximise the success rate and searching paths of the directory.
7/4	Informative feedback as to the progress of the search process shall always be displayed.
7/5	Feedback from the user shall be obtained to assist in narrowing the search options in cases of multiple choices. A list of options should be presented to the user.
7/6	User shall be able to cancel the search process at any stage.
7/7	If search is successful, the recipients O/R address is read and automatically stored with an associated nickname supplied from the user.
UA Administration Recommendations	
8/1	A list of deferred delivery messages shall be available for review. The messages can be read and modified, deleted, held indefinitely, or sent immediately.
8/2	Messages can be automatically forwarded to another recipient. This toggle button is off by default. An option menu of nicknames is used to specify the recipient. Other optional information (heading and comment fields) are entered into text fields.
8/3	Message receipt notifications can be automatically generated. This toggle button should be on by default. Supplementary information is entered into a text field.
8/4	Two toggle buttons indicate if date-expired and obsoleted messages are to be automatically removed. The default is true.
8/5	Management of message folders and folder groups (creation, removal, and modification) shall be supported. A list of message folders can be used to edit the names.

Number	IRM Recommendation
8/6	Nickname management (creation, removal, and modification) shall be provided. The O/R address entry screen shall be used to modify existing nicknames.
O/R Address Layout Recommendations	
9/1	The printing of O/R addresses (eg on business cards) should follow a vertical layout. Each attribute/value pair should be displayed with the acronym and value.

Cited Works

Abowd, Gregory & Bowen, Jonathan & Dix, Alan & Harrison, Michael & Took, Roger. *User Interface Languages: A Survey of Existing Methods.* Oxford University Computing Laboratory Research Report PRG–TR–5–89. 1989.

Adams, Dennis A & Todd, Peter A & Nelson, R Ryan. *A comparative evaluation of the impact of electronic and voice mail on organizational communication*. **Information & Management** 24 (1993): 9–21.

Afifi, Hossam & Huitema, Christian. *Solving Names within X.500*. INRIA Research Report 1633, France. 1992.

Akin, Omer & Rao, D Radha. *Efficient computer-user interface in electronic mail systems*. **International Journal of Man-Machine Studies** 22 (1985): 589–611.

Allocchio, Claudio & Ghiselli, Antonia. *The INFN GIVEME 987 Gateway*. **Computer Networks and ISDN Systems** 19 (1990): 255–260.

Altmare, G & Berlen, O & Rotondi, D & Scopece, A. *DICOM: An Approach to the MHS UA design*. **Message Handling Systems and Distributed Applications**. Stefferud *et al*, eds. Elsevier Science Publishers (1989): 415–425.

Alvestrand, Harald T. *Frequently Asked Questions Product list—Products implementing the X.400 standards.* Revision 1.12, News Group: comp.protocols.iso.x400 (1 June 1993).

Anderson, Bob & Heath, Christian & Luff, Paul & Moran, Thomas P. *The Social and the Cognitive in Human-Computer Interaction.* Technical Report EPC-91-126, Rank Xerox, Cambridge UK, 1991.

Ankrah, Anne & Frohlich, David M & Gilbert, G Nigel. *Two Ways to Fill a Bath, with and without knowing it.* **Human-Computer Interaction—INTERACT'90.** Diaper, D *et al*, eds. Elsevier Science Publishers. (1990): 73–77.

Apple Computer. *Human Interface Guidelines: The Apple Desktop Interface.* Addison-Wesley, 1987.

Arbour, Nancy H. *The Design of Human-Computer Interfaces*. Master of Science Thesis. Department of Operations Research and Information Systems, Eastern Michigan University, 1990.

Artim, John & Hary, Joseph & Spickhoff, Franz. *User interface services in AD/Cycle*. **IBM Systems Journal** 29.2 (1990): 236–249.

Australian Personal Computer. *From TTY to VUI*. **Australian Personal Computer** (May 1990): 161–170.

Backer, Andreas & Genau, Andreas. *GINA++ The Generic Interactive Application for C++ and OSF/Motif User Manual.* German National Research Centre for Computer Science. Feb 1992.

Barfield, Lon. *The User Interface Concepts & Design*. Addison-Wesley, 1993.

Barker, H A & Chen, M & Grant, P W & Jobling, C P & Parkman, A & Townsend, P. *The use of OPENLOOK/Motif GUI Standards for Applications in Control System Design*. **User Interface Management and Design**. Duce, D A, *et al*, eds. Springer-Verlang (1990): 295–305.

Barker, Paul. *Managing data derived from multiple sources in an X.500 Directory*. **Computer Communications Review** (Jan 1991): 61–70.

Barker, Paul. *A Public Access Interface to the OSI Directory*. **Proceedings of EurOpen Autumn '91**, Budapest (Sept 1991): 173–186.

Barker, P. *DUA Metrics*. **OSI-Directory Services Internet Draft 33**, June 1992.

Barker, Paul. *Implementing archive retrieval using X.500*. **OSI-Directory Services Internet Draft 40**, March 1993.

Barker, P & Hardcastle-Kille, S E. *Naming Guidelines for Directory Pilots*. **OSI-Directory Services Internet Draft 12**, April 1992.

Barker, P & Hardcastle-Kille, S E. *DSA Metrics*. **OSI-Directory Services Internet Draft 34**, June 1992.

Barker, P & Kille, S. *The COSINE and Internet X.500 Schema*. **Request for Comments 1274**, Nov 1991.

Beck, A & Ziegler, J. *New Approaches in Software Engineering for Interactive Systems.* **Human Aspects in Computing: Design and Use of Interactive Systems and Work with Terminals.** Bullinger, H J, ed. Elsevier Science Publishers (1991): 783–787.

Becker, Phil. *The Psychology of Electronic Mail*. **Boardwatch Magazine** (Oct 1990).

Benbasat, Izak & Todd, Peter. *An Experimental Investigation of Interface Design Alternatives: Icon vs Text and Direct Manipulation vs Menus*. **International Journal Man-Machine Studies** 38 (1993): 369–402.

Benimoff, Nicholas I & Whitten, Williams B. *Human Factors Approaches to Prototyping and Evaluating User Interfaces.* **AT&T Technical Journal** (Sept/Oct 1989): 44–55

Bennett, John L. *Collaboration of UIMS Designers and Human Factors Specialists*. **Computer Graphics** 21.2 (Apr 1987): 102–105.

Benyon, D & Murray, Dianne. *Experiences with Adaptive Interfaces.* **The Computer Journal** 31.5 (1988): 465–473.

Berry, R E. *Common User Access—A consistent and usable human-computer interface for the SAA environments.* **IBM Systems Journal** 27.3 (1988): 281–300.

Berry, R E & Reeves, C J. *The evolution of the Common User Access Workplace Model.* **IBM Systems Journal** 31.3 (1992): 414–428.

Betts, Bill & Burlingame, David & Fischer, Gerhard & Foley, Jim & Green, Mark & Kasik, David & Kerr, Stephen T & Olsen, Dan & Thomas, James. *Goals and Objectives for User Interface Software.* **Computer Graphics** 21.2 (Apr 1987): 73–78.

Bevan, Nigel. *Human Factors in the Development of a Standard for X.400 Electronic Mail*. **Proceedings of CHI91** Interactive Poster Session (1991): 493.

Bevan, Nigel. *Standards Relevant to European Directives for Display Terminals.* **Human Aspects in Computing: Design and Use of Interactive Systems and Work with Terminals.** Bullinger, H J, ed. Elsevier Science Publishers (1991): 533–537.

Bijendra, Jain N & Agrawala, Ashok K. *Open Systems Interconnection: Its Architecture and Protocols*. Elsevier Science Publishers, 1990.

Bishop, Matt. *Privacy-enhanced Electronic Mail*. **Internetworking: Research and Experience** 2 (1991): 199–233.

Black, Uyless D. *OSI: A Model for Computer Communications Standards*. Prentice Hall Inc, 1991.

Blum, Daniel. *An Analysis and Comparison of Internet and X.400 Messaging*. **ConneXions** 6.9 (Sept 1992): 40–52.

Blum, Daniel & Rowe, Gary. *E-mail switches emerge as enterprising idea*. **Network World** (May 1993): 68–76.

Bodker, Susanne. *Through the Interface: A Human Activity Approach to User Interface Design.* New Jersey: Lawrence Erlbaum Associates, 1991.

Bonsiepe, Gui. *Interpretations of Human User Interface*. **Visible Language** 24.3–4 (1990): 262–285.

Borenstein, Nathaniel S. *Multimedia Mail from the Bottom Up*. **ConneXions** 5.11 (Nov 1991): 10–16.

Borenstein, N & Freed, N. *MIME (Multipurpose Internet Mail Extensions): Mechanisms for Specifying and Describing the Format of Internet Message Bodies.* **Request For Comments 1341**, June 1992.

Borras, P & Mamou, J C & Plateau, D & Poyet, B & Tallot, D. *Building user interfaces for database applications: The O_2 experience.* **SIGMOD Record** 21.1 (Mar 1992): 32–38.

Brackeen, Debra & Grits, Deborah & Mackraz, Barbara & Tognazzini, Bruce. *Apple Human Interface Checklist*. Apple Computer Inc. 1989.

Brown, C Marlin. *Human-Computer Interface Design Guidelines.* Ablex Publishing, 1988.

Brown, J R & Cunningham, S. *Programming the User Interface.* New York: John Wiley & Sons, 1989.

Burns, Nina & Radicati, Sara. *Breaking Through*. **Corporate Computing** 1.1 (1993): 166–184.

Capucciati, Maria R. *Putting Your Best Face Forward: Designing an Effective User Interface.* **Microsoft Systems Journal.** (Feb 1993): 55–63.

Carey, Tom & Crentsil, Joseph. *Integrating Widget Design Knowledge with User Interface Toolkits*. **Proceedings CASE'92**. (1992): 204–212.

CCITT. *Data Communications Networks: Message Handling Systems—Recommendations X.400–X.420*. Blue Book Volume VIII—Fascicle VIII.7. The International Telegraph and Telephone Consulative Committee, 1988.

CCITT. *Data Communications Networks: Directory—Recommendations X.500–X.521*. Blue Book Volume VIII—Fascicle VIII.8. The International Telegraph and Telephone Consulative Committee, 1988.

CCITT. *MHS Implementors' Guide*. Version 8. CCITT Special Rapporteur's Group on Message Handling Systems (Question 18/VII) and ISO/IEC JTC 1/SC 18/WG 4 SWG on Messaging. March 1992.

CCITT. *Representation of O/R Addresses for Human Usage*. **CCITT F.401/ISO 10021-2**, Annex F, Feb 1993.

Chapman, Peter. *X.400 & The User Interface*. **Proceedings of Electronics Message Systems**. Online Publications, UK (1986): 105–127.

Chen, Jolly. *Providing Intrinsic Support for User Interface Monitoring*. **Human-Computer Interaction—INTERACT'90**. Diaper, D, *et al*, eds. Elsevier Science Publishers (1990): 415–420.

Chignell, Mark H. *A Taxonomy of User Interface Terminology.* **SIGCHI Bulletin** 21.4 (April 1990): 27–34.

Chilton, P A. X400: *The messaging and interconnection medium for the future*. Manchester: NCC Blackwell Ltd, 1990.

Chimera, Richard. *Value Bars: An Information Visualisation and Navigation Tool for Multi-attribute Listings and Tables.* University of Maryland Technical Report CAR-TR-589. Oct 1991.

Cini, A. *Mail Architectures*. **DEC Professional** (March 1990): 108–114.

Cockton, G & Harrison, M & Kwasnik, B & Penner, R & Procter, R. *HCI: Whose problem is it anyway?* **Proceedings of IFIP TC2/WG2.7 Working Conference**, Finland, 10–14 Aug, 1992.

Cole, Robert & Burns, John. *An Architecture for a Mobile OSI Mail Access System*. **IEEE Journal on Selected Areas in Communications** 7.2 (Feb 1989): 249-256.

Colville, John. *An X-Windows Manager for the Directory Service*. Key Centre for Advanced Computing Sciences Research Report 91.16, University of Technology, Sydney, Australia, 1991.

Cox, Kevin & Walker, David. *User-Interface Design*. Advanced Education Software, 1990.

Crellin, Jonathan & Horn, Thomas & Preece, Jenny. *Evaluating Evaluation: A Case Study of the use of Novel and Conventional Evaluation Techniques in a Small Company*. **Human-Computer Interaction—INTERACT'90**. Diaper, D, *et al*, eds. Elsevier Science Publishers (1990): 329–335.

Crocker, David H. *Standard for the format of ARPA Internet text messages.* **Request For Comments 822**, Aug 1982.

Dance, John R & Granor, Tamar E & Hill, Ralph D & Hudson, Scott E & Meads, Jon & Myers, Brad A & Schulert, Andrew. *The Run-time Structure of UIMS-Supported Applications.* **Computer Graphics** 21.2 (Apr 1987): 97–101.

Danielsen, Thore & Finnset, Wiggo & Flaegstad, Frode & Hartvigsen, Gunnar & Steen, Roar. *Ratatosk—An Adaptive User Interface to Computer-Mediated Communication*. **Message Handling Systems and Application Layer Communication Protocols.** Schicker, P & Stefferud, E, eds. Elsevier Science Publishers (1991): 345-354.

Day, Mary Carol. *Designing the Human Interface: An Overview.* **AT&T Technical Journal** (Sept/Oct 1989): 2–8.

Del Galdo, E. *Internationalization and Translations: Some Guidelines for the Design of HCI.* **Designing User Interfaces for International Use**. Nielsen, J ed. Amsterdam: Elsevier Science Publications (1990): 1–10.

Denning, Peter J & Dargan, Pamela A. *A Discipline of Software Architecture*. **Interactions** 1.1 (Jan 1994): 55–65.

De Waal, Benny M E & Van Der Heiden, Gerard H. *The Evaluation of User-Friendliness in the Design Process of User Interfaces*. **Human Factors in Information Systems Analysis and Design**. Finkelstein, A & Tauber, M J & Traunmuller, R, eds. Elsevier Science Publishers (1990): 93–103.

Diaper, Dan. *Task Analysis for Human-Computer Interaction*. Ellis Horwood Ltd, 1989.

Dickson, Gary & Lloyd, Alan. *Open Systems Interconnection*. Prentice Hall Australia, 1992.

Dowell, John & Long, John. *A 'Late' Evaluation of a Messaging System Design and the 'Target' of 'Early' Evaluation Methods*. **People and Computers 5**. Sutcliffe, A & Macaulay, L, eds. Cambridge University Press (1989): 331–344.

Draper, Stephen W & Norman, Donald A. *Software Engineering for User Interfaces*. **IEEE Transactions on Software Engineering** 11.3 (Mar 1985): 252–258.

Dreyfuss, Henry. *Symbol Source Book: An authoritative Guide to International Graphics Symbols.* New York: McGraw-Hill, 1972.

Duffy, Thomas M & Palmer, James E & Mehlanbacher, Brad. *Online Help: Design and Evaluation*. New Jersey: Ablex Publishing Corporation, 1992.

Dunfee, William & McGehe, J & Rauf, Robert & Shipp, K. *Designing SAA applications and user interface*. **IBM Systems Journal** 27.3 (1988): 325–346.

Eason, Ken & Harker, Susan. *Institutionalising Human Factors in the Development of Teleinformatics Systems*. **Research into Networks and Distributed Applications** Speth, R, ed. Elsevier Science Publishers (1988): 15–25.

Eaves, Dan. *Costing Computer System Interfaces: An Enterprise Perspective*. Working Paper 1/91. Monash University, Australia. 1991.

Edmonds, Ernest. *The emergence of the separable user interface*. **ICL Technical Journal** (May 1990): 54–65.

Eglowstein, Howard & Smith, Ben. *Mixed Messaging*. **BYTE** (March 1993): 136–154.

Encarnacao, Jose & Cote-Munoz, Jairo & Eckardt, Dieter & Rix, Joachim & Teixeira, Jose. *User interfaces to support the design process.* **Computers in Industry** 17 (1991): 317–333.

Erickson, Tom. *Duelling Metaphors: the Desktop & HyperCard*. **Human Interface Notes** Number 3. Apple Computer Inc. Jan 1990.

Exner, Rolf. X.*500 Electronic Directories*. **Naming and Addressing in the OSI Environment**, Standards Australia Seminar. (Sept 1992): 38–46.

Farber, James M. *The AT&T User-Interface Architecture*. **AT&T Technical Journal.** (Sept/Oct 1989): 9–16.

Farris, Rick. *UNIX Mail Gets Easier*. **Unix World** (Nov 1991): 111–115.

Fekete, Jean-Daniel. *WWL Widget Wrapper Library for C++ Reference Manual*. Laboratoire de Recherche en Imformatique, France. Feb 1991.

Fischer, Gerhard. *An Object-oriented Construction and Tool Kit for Human-Computer Communication.* **Computer Graphics** 21.2 (Apr 1987): 105–109.

Fischer, Gerhard. *Human-Computer Interaction Software: Lessons Learned, Challenges Ahead*. **IEEE Software** (Jan 1989): 44–52.

Foley, James. *Transformations on a Formal Specification of User-Computer Interfaces.* **Computer Graphics** 21.2 (Apr 1987): 109–113.

Foley, James & Kim, Won Chul & Kovacevic, Srdjan & Murray, Kevin. *Defining Interfaces at a High Level of Abstraction.* **IEEE Software**. (Jan 1989):25–32.

Foody, Michael. *UIMX: A User Interface Management System for Scientific Computing with X Windows*. **SPIE Vol 1083 Three-Dimensional Visualization and Display Technologies** (1989): 228–233.

Forsdick, Harry & Thomas, Robert & Robertson, George & Travers, Virginia. *Initial Experience with Multimedia Documents in Diamond*. **Computer-Based Message Services**. Smith, H T, ed. Elsevier Science Publishers (1984): 99–113.

Fox, Jeffery A. *DRUID V2.0 Users Manual.* The MITRE Corporation, Bedford, MA 01730-0208, USA. 1991.

Galitz, Wilbert O. *User-Interface Screen Design.* Boston: QED Publishing Group, 1993.

Garcia-Luna-Aceves, J J. *Towards Computer-Based Multimedia Information Systems*. **Computer Message Systems–85**. Uhlig, R, ed. Elsevier Science Publishers (1986): 61–77.

Garcia-Luna-Aceves, J J & Poggio, A. *Multimedia Message Content Protocols For Computer Mail*. **Computer-Based Message Services**. Smith, H T, ed. Elsevier Science Publishers (1984): 87–99.

Genilloud, Guy. *X.400 MHS: First Steps Towards an EDI Communication Standard*. **Computer Communication Review** 20.2 (April 1990): 72–86.

Gerlach, James H & Kuo, Feng-Yang. *Formal Development of Hybrid User-Computer Interfaces with Advanced Forms of User Assistance.* **Journal of Systems Software** 16 (1991): 169–183.

Gillooly, Caryn. *Study says X.400 unlikely to gain ground*. **ComputerWorld Australia**. Communications World Supplement (Aug 1991): 7.

GO Corporation. *PenPoint™ User Interface Design Reference*. Addison-Wesley, 1992.

Good, M & Whiteside, J &Wixon, D & Jones, S. *Building a User-Derived Interface.* **Communications of the ACM** 27.10 (Oct 1984): 1032–1043.

Gorny, Peter. *Software Engineering = Human Factors Engineering?* **Human Aspects in Computing: Design and Use of Interactive Systems and Work with Terminals.** Bullinger, H J, ed. Elsevier Science Publishers (1991): 551–555.

Gould, John D & Lewis, Clayton. *Designing for Usability: Key Principles and What Designers Think*. **Communications of the ACM** 28.3 (Mar 1985): 300–311.

Gould, John D & Boies, Stephen J & Levy, Stephen & Richards, John T & Schoonard, Jim. *The 1984 Olympic Message System: A Test of behavioural Principles of System Design.* **Human-Computer Interaction**: **Selected Readings.** Preece, Jenny & Keller, Laurie & Stolk, Hans, eds. Hertfordshire, UK: Prentice-Hall, 1990: 260–283.

Goyer, P & Radureau, J. *GENEPX400—A Test System for X.400 Based Message Handling Systems*. **Computer Standards & Interfaces** 7 (1988): 103–109.

Green, Mark. *A Survey of Three Dialogue Models*. **ACM Transactions on Graphics** 5.3 (July 1986): 244–275.

Greene, Nancy & Tissot, Roland. *A Distributed User Agent: Concept and Implementation*. **Message Handling Systems**. Speth, R, ed. Elsevier Science Publishers (1988): 345–354.

Grimm, Rudiger & Heagerty, Denise. *Recommendation for a Shorthand X.400 Address Notation*. **Computer Networks and ISDN Systems** 17 (1989): 263–267.

Grudin, Jonathan. *The Case Against User Interface Consistency.* **Communications of the ACM** 32.10 (Oct 1989): 1164-1173.

Grudin, Jonathan. *Consistency, Standards, and Formal Approaches to Interface Development and Evaluation: A Note on Wiecha, Bennett, Boies, Gould, and Greene.* **ACM Transactions on Information Systems** 10.1 (Jan 1992): 103-111.

Hamer, Malcolm & Heilmann, Jim. *How one firm created its own global electronic mail network.* **Data Communications** (June 1988): 167–184.

Hammond, Judy & McManus, Brendan. *Integrating Usability Evaluation with Systems Design.* **Human Aspects in Computing: Design and Use of Interactive Systems and Work with Terminals.** Bullinger, H J, ed. Elsevier Science Publishers (1991): 733–737.

Hansen, Alf. *Coordination of Mail Gateways: The X.400–RFC Problem.* **Computer Networks and ISDN Systems** 19 (1990): 251–254.

Hansen, Alf. *THE 400: Internet X.400 Pilot Project Newsletter*. Number 2, May 17, 1991.

Happ, Alan J & Stanners, Sharon L & Keller, Arthur. *An Evaluation of Alternative Designs for Variable Selection Lists in the IBM CUA Basic Interface Design Guide.* Technical Report TR54.593. IBM, Florida. 1991.

Hardcastle-Kille, S E. *Mapping between X.400 (1988) / ISO 10021 and RFC 822.* **Request For Comments 1148bis**, May 1991.

Hardcastle-Kille, S E. *Replication and Distributed Operations extensions to provide an Internet Directory using X.500*. **Request for Comments 1276**, Nov 1991.

Hardcastle-Kille, S E. *Encoding Network Addresses to support operation over non-OSI layers*. **Request for Comments 1277**, Nov 1991.

Hardcastle-Kille, S E. *X.500 and Domains*. **Request for Comments 1279**, Nov 1991.

Hardcastle-Kille, S E. *Using the OSI Directory to achieve User Friendly Naming*. **OSI-Directory Services Internet Draft 24**, Jan 1992.

Hardcastle-Kille, S E. *Representing the Real World in the OSI Directory*. **OSI-Directory Services Internet Draft 25**, Feb 1992.

Hardcastle-Kille, S E. *The QUIPU Directory Implementation*. **ConneXions** 6.9 (Sep 1992): 10–15.

Hardcastle-Kille, S E. *The PP Message Transfer Agent*. **ConneXions** 6.9 (Sep 1992): 16–21.

Hardcastle-Kille, S & Huizer, E & Cerf, V & Hobby, R & Kent, S. *A Strategic Plan for Deploying an Internet X.500 Directory Service*. **Request for Comments 1430**, Feb 1993.

Hartson, Rex. *User-Interface Management Control and Communication*. **IEEE Software** (Jan 1989): 63–70.

Hartson, H Rex & Hix, Deborah. *Human-Computer Interface Development: Concepts and Systems for its Management.* **ACM Computing Surveys** 21.1 (March 1989): 5–92.

Hartson, H Rex & Hix, Deborah. *Toward empirically derived methodologies and tools for human-computer interface development.* **International Journal Man-Machine Studies** 31 (1989): 477–494.

Hartson, H Rex & Siochi, Antonio C & Hix, Deborah. *The UAN: A User-Oriented Representation for Direct Manipulation Interface Designs.* **ACM Transactions on Information Systems** 8.3 (Jul 1990): 181–203.

Hartson, H Rex & Smith, Eric C. *Rapid prototyping in human-computer interface development.* **Interacting with Computers** 3.1 (1991): 51–91.

Hayes, Frank. *Going Graphical*. **UnixWorld** (July 1992): 46–52.

Hayes, Frank. *Rivals Compete for E-mail Standards*. **UnixWorld** (July 1992): 89–90.

Henken, Gerrit. *Mapping of X.400 and RFC822 Addresses*. **Computer Networks and ISDN Systems** 13 (1987): 161-164.

Henken, Gerrit. *Development and Interconnection of X.400 Message Handling Systems*. **Computer Standards & Interfaces** 7 (1988): 17–22.

Henshall, John & Shaw, Sandy. *OSI Explained: End-to-End Computer Communication Standards* 2nd Ed. Ellis Horwood Limited, 1990.

Hesketh, Richard. *Perly—UNIX with Buttons*. **Software-Practice and Experience**. 21.11 (Nov 1991): 1165–1187.

Hicks, Richard & Essinger, James. *Making computers more human: Designing for human-computer interaction.* Oxford: Elsevier Science Publishers, 1991.

Hinrichs, Elke & Prinz, Wolfgang. *Design Examples of a Window-based X.500 User Agent Interface*. **Message Handling Systems and Application Layer Communication Protocols**. Schicker, P & Stefferud, E, eds. Elsevier Science Publishers (1991): 369–388.

Holleran, Patrick A & Haller, Richard W. *Characteristics of Well-Designed Electronic Communications Systems*. **Human-Computer Interaction—INTERACT'90**, Diaper, D *et al*, eds. Elsevier Science Publications (1990): 841–847.

Hong, James W & Bauer, Michael A & Bennett, J Michael. *Integration of the Directory Service in the Network Management Framework*. **Proceedings of Third International Symposium on Integrated Network Management**, San Fransisco (Apr 1993): 149–160.

Houldsworth, Jack. *Applying Electronic Messaging*. **Telecommunications** (Aug 1990): 61–73.

Howes, Timothy A. *The Gopher to X.500 Gateway*. **ConneXions** 7.4 (Apr 1993): 12–18.

Howes, T & Smith, M & Beecher B. *DIXIE Protocol Specification*. **Request for Comments 1246**, July 1991.

Hu, Weiming. *Making ends meet: Interconnecting electronic mail networks*. **Data Communications** (Sept 1988): 128–141.

Hudson, Scott E. *UIMS Support for Direct Manipulation Interfaces.* **Computer Graphics** 21.2 (Apr 1987): 120–124.

Hudson, Scott E & Mohamed, Shamim P. *Interactive Specification of Flexible User Interface Displays*. **ACM Transactions on Information Systems** 8.3 (Jul 1990): 269–288.

Hunt, Ray. *CCITT X.500 Directories—principles and applications*. **Computer Communications** 15.10 (Dec 1992): 636–645.

Hurley, William D. & Sibert, John L. *Modeling User Interface-Application Interactions.* **IEEE Software**. (Jan 1989): 71–77.

Hutchins, Edwin L & Hollan, James D & Norman, Donald A. *Direct Manipulation Interfaces.* **User Centered System Design.** Chapter 5. Norman, Donald A & Draper, Stephen W, eds. New Jersey: Lawrence Erlbaum Associates, 1986.

Hutt, Andrew T F & Flower, Fiona. *The User System Interface—a challenge for application users and application developers?* **ICL Technical Journal**. (May 1990): 43–53.

Iannella, Renato. *Designing 'Safe' User Interfaces.* **Proceedings of Australian Software Engineering Conference**. Sydney. (July 1991): 177–188.

Iannella, Renato. *BRUIT∆SAM—An Interface for User Interface Guidelines.* **Proceedings of Australian Computer Society Queensland Branch Conference**. Gold Coast, Australia. 3–5th April, 1992.

Iannella, Renato. *Directory Systems Interfaces*. **Proceedings of OZCHI92**, Rees, M J & Iannella, R, eds. Gold Coast, Australia, 26–27 Nov (1992): 110-117.

Intergraph Corporation. *MicroStation Mac Users Guide.* USA, August 1989.

ISO/IEC. *Information Technology—Text and Office Systems—Graphical Symbols Used on Screens: Interactive Icons*. Draft International Standard JTC 1/SC 18 N 3737. International Standards Organisation (ISO) & International Electrotechnical Commission (IEC), July 1992.

Israelski, Edmond W & Angiolillo-Bent, Joel S & Brems, Douglas J & Hoag, LaVerne L & Roberts, Linda A & Wells, Robin S. *Generalizable User-Interface Research*. **AT&T Technical Journal** (Sept/Oct 1989): 31-43.

Jacob, Robert J K. *Using Formal Specifications in the Design of a Human-Computer Interface.* **Communications of the ACM** 26.4 (April 1983): 259–264.

Jacob, Robert J K. *The Use of Eye Movements in Human-Computer Interaction Techniques: What You Look At Is What You Get*. **ACM Transactions on Information Systems** 9.3 (Apr 1991): 152–169.

Jakobs, Kai & Karabek, Rashid. *Bringing Application Services and Users Together—The Electronic Mail Example*. **Proceedings of the International Conference on Private Switching Systems**. IEE, London (2 June 1992): 199–205.

James, P & Churcher, J. *Cosmos—a project in structured messaging*. **British Telecom Technology Journal** 6.4 (October 1988): 37–43.

Jeffree, Tony. *X.400: The naming and addressing problem*. **Proceedings of Electronic Messaging and Communications Systems**. London: Blenheim Online Publications (1989): 123–134.

Jenson S & Sullivan J. *Pop-Up Menus*. **Human Interface Notes** Number 9. Apple Computer Inc. Oct 1990.

Jorgensen, Anker Helms. *Using the Thinking-Aloud Method in System Development*. **Designing and Using Human-Computer Interfaces and Knowledge Based Systems**. Salvendy, G & Smith, M J, eds. Elseivier Science Publishers (1989): 743–750.

Kantorowitz, Eliezer & Sudarsky, Oded. *The Adaptable User Interface*. **Communications of the ACM** 32.11 (Nov 1989): 1352–1358.

Kairi, Kalliopi. *Design and Implementation of an X.400 Stand-Alone User Agent*. Master of Computing Science Thesis, Queen's University, Canada, April 1987.

Kairi, Kalliopi & Barnard, David T. *Design and Implementation of an X.400 Stand-Alone User Agent*. **Computer Standards & Interfaces** 7 (1988): 219–232.

Karat, Clare-Marie. *Usability Engineering in Dollars and Cents*. **IEEE Software** 10.3 (May 1993): 88–89.

Kasik, David J & Lund, Michelle A & Ramsey, Henry W. *Reflections on Using a UIMS for Complex Applications*. **IEEE Software** (Jan 1989): 54–61.

Katsaros, John. *Implementing an X.400-Based Corporate Mail System*. **Telecommunications**. (Nov 1990): 49–54.

Katz, Alan R. *An Experimental Internetworking Multimedia Mail System*. **Computer-Based Message Services**. Smith, H T, ed. Elsevier Science Publishers (1984): 115–123.

Kille, Steve. *Integration of Electronic Mail and Conferencing Systems*. **Computer-Based Message Services**. Smith, H T, ed. Elsevier Science Publishers (1984): 261–269.

Kille, Steve. *Integration of Message Handling and Directory: A System Manager's View*. **Proceedings of EUUG Autumn '90**, Nice (Oct 1990): 203–207.

Kille, Stephen E. *Implementing X.400 and X.500: The PP and QUIPU Systems*. Boston: Artech House Inc, 1991.

Kille, Steve & Onions, Julian. *The PP Manual*. University College London & X-Tel Services Ltd, 1991.

King, Julia. *A Mailbox Muddle*. **Communications Networks** (Jan 1992): 43–47.

King, Steven S. *Message-Delivery APIs: The Message is the Medium*. **Data Communications** (April 1992): 85–95.

Kishi, Nobuko & Kinoe, Yosuke. *Assessing Usability Evaluation Methods in a Software Development Process*. **Human Aspects in Computing: Design and Use of Interactive Systems and Work with Terminals.** Bullinger, H J, ed. Elsevier Science Publishers (1991): 597–601.

Kobara, Shiz. *Visual Design with OSF/Motif*. Addison-Wesley, 1991.

Koorland, Neil. *The OSI and X.400 APIs—An Overview*. **Message Handling Systems and Application Layer Communication Protocols**. Schicker, P & Stefferud, E, eds. Elsevier Science Publishers (1991): 15–22.

Koritzinsky, Ianne Howards. *New Ways to Consistent Interfaces*. **Coordinating User Interfaces for Consistency**, Nielsen, Jakob, ed. Academic Press (1989): 93–106.

Kuhme, Thomas & Schneider-Hufschmidt, Matthias. *SX/Tools—An Open Design Environment for Adaptable Multimedia User Interfaces*. **Eurographics'92** 11.3, Kilgour, A & Kjelldahl, L, eds. Blackwell Publishers (1992): 93–105.

Kuo, Feng-Yang. *An Object-Oriented Approach to the Design of a Mail System for a Heterogeneous Environment*. **Information & Management** 15 (1988): 173-182.

Laborie, Bernard & Huitema, Christian. *X.400: Towards Global Connectivity*. **Message Handling Systems**. Speth, R, ed. Elsevier Science Publishers (1988): 23–34.

Lafuente, Juan M. *High-level language extensions for user-interface programming.* **Software Engineering Journal**. (Nov 1992): 409–420.

Lane, Thomas G. *A Design Space and Design Rules for User Interface Software Architecture*. Software Engineering Institute, Technical Report CMU/SEI–90–TR–22. Nov 1990.

Lane, Thomas G. *User Interface Software Structures*. Doctoral Thesis. School of Computer Science, Carnegie Mellon University, 1990.

Lang, Ruth & Wright, Russ. *A Catalog of Available X.500 Implementations*. **Directory Information Services Internet Draft**, July 1991.

Lantz, Keith A. *Multi-process Structuring of User Interface Software.* **Computer Graphics** 21.2 (Apr 1987): 124–130.

Lantz, Keith A & Tanner, Peter P & Binding, Carl & Huang, Kuan-Tsae & Dwelly, Andrew. *Reference Models, Window Systems, and Concurrency.* **Computer Graphics** 21.2 (Apr 1987): 87–97.

Laurel, Brenda. *Interface as Mimesis.* **User Centered System Design.** Chapter 4. Norman, Donald A & Draper, Stephen W, eds. New Jersey: Lawrence Erlbaum Associates, 1986.

Law, Carl Edgar. *X.400 and OSI Electronic Messaging into the 1990s*. London: IBC Technical Services Ltd, 1989.

Lewis, Clayton. *Using the 'Thinking-aloud' Method in Cognitive Interface Design.* Research Report RC 9265 (#40713). IBM, New York. 1982.

Lewis, Clayton & Norman, Donald A. *Designing for Error.* **User Centered System Design.** Chapter 20. Norman, Donald A & Draper, Stephen W, eds. New Jersey: Lawrence Erlbaum Associates, 1986.

Lewis, T G & Handloser III, Fred & Bose, Sharada & Yang, Sherry. *Prototypes from Standard User Interface Management Systems.* **IEEE Computer** (May 1989): 51–60.

Lewis, Chris. *UNIX EMail Software—a Survey*. Frequently Asked Questions mailfaq@ferret.ocunix.on.ca, 1992.

Liao, Holmes S & Osada, Masakazu & Shneiderman, Ben. *A Formative Evaluation of Three Interfaces for Browsing Directories Using Dynamic Queries.* University of Maryland Technical Report CAR-TR-605. Feb 1992.

Lindgaard, Gitte & Bednall, Elizabeth & Chessari, Josephine. *Improving User Performance*, **IEEE Journal on Selected Areas in Communications** 9.4 (May 1991): 506–517.

Linn, J. *Privacy Enhancement for Internet Electronic Mail: Part I—Message Encipherment and Authentication Procedures*. **Request For Comments 1113**, August, 1989.

Linton, Mark A & Vlissides, John M & Calder, Paul R. *Composing User Interfaces with InterViews*. **Computer** (Feb 1989): 8–22.

Luff, Paul & Heath, Christian. *The Practicalities of Menu Use: Improvisation in Screen-Based Activity.* Rank Xerox EuroPARC Technical Report EPC-91-102.1. Cambridge, 1991.

Lund, Arnold M & Tschirgi, Judith E. *Designing for People: Intergrating Human Factors into the Product Realization Process.* **IEEE Journal On Selected Areas In Communications** 9.4 (May 1991): 496–500.

Lung, Loh Wai & Swee, Poo Gee. *An Object-Oriented Approach to Message Handling System*. **Proceedings Singapore International Conference on Networks**. Poo, G S, ed. (5–6 Sept 1991): 25–29.

Lutz, Ernst & Kleist-Retzow, Hans & Hoernig, Karl. *MAFIA—An Active Mail-Filter-Agent for an Intelligent Document Processing Support*. **Multi-User Interfaces and Applications**. Gibbs, S & Verrijn-Stuart, A A, eds. Elsevier Science Publishers (1990): 235–251.

MacIntyre, Blair. *A Constraint-based Approach to Dynamic Colour Management for Windowing Interfaces*. Masters Thesis, University of Waterloo, Canada 1991.

Mackinlay, Jock D & Robertson, George G & Card, Stuart K. *The Perspective Wall: Detail and Context Smoothly Integrated*. **Proceedings of CHI'91,** New Orleans, USA, April 27–May 2 (1991): 173–179.

MacLeod, Miles. *Direct Manipulation Prototype User Interface Monitoring*. **People and Computers 5**. Sutcliffe, A & Macaulay, L, eds. Cambridge University Press (1989): 395–407.

Mahl, Damanjit. *A Design Overview of XLookUp*. **Proceedings of EurOpen Autumn '91**, Budapest (Sept 1991): 199–211.

Magedanz, Thomas & Tschichholz, Michael & Weiss, Karl-Heinz. *Interconnection of X.400 Systems and notable Gateways*. **Message Handling Systems**. Speth, R, ed. Elsevier Science Publishers (1988): 3–21.

Maguire, M C. *A Review of Human Factors Guidelines and Techniques for the Design of Graphical Human-Computer Interfaces*. **Human-Computer Interaction**: **Selected Readings**, Preece, Jenny & Keller, Laurie, & Stolk, Hans, eds. Prentice Hall (1990): 161–184.

Malone, Thomas W & Grant, Kenneth R & Turbak, Franklyn A. *The Information Lens: An Intelligent System for Information Sharing in Organisations*. **Proceedings of CHI'86** (1986): 1–8.

Mantei, Marilyn M & Teorey, Toby J. *Cost/Benefit Analysis for Incorporating Human Factors in the Software Lifecycle*. **Communications of the ACM** 31.4 (Apr 1988): 428–439.

Mantelman, Lee. *A scheme to protect open networks' security*. **Data Communications International** (Aug 1989): 37–39.

Marcus, Aaron. *The Ten Commandments of Color: A Tutorial*. **Computer Graphics Today** 3.10 (Nov 1986): 7+.

Marcus, Aaron. *Designing Graphical User Interfaces*. **Unix World**. Part 1 (Aug 1990): 107–115, Part 2 (Sep 1990): 121–127, Part 3 (Oct 1990): 135–138.

Marcus, Aaron. *Graphic Design for Electronic Documents and User Interfaces*. New York: Addison-Wesley, 1992.

Marine, A. *X.500 Pilot Projects*. **Directory Information Services Internet Draft**, June 1993.

Marshak, David S. *Name Services*. **Network Monitor** 6.8, (1991): 3–20.

Marshak, David S. *The Quest for Common Mail APIs*. **Office Computing Report** 15.8 (1992): 3–21.

Marshak, David S. *The New E-Mail APIs*. **Distributed Computing Monitor** 7.7 (1992): 3–26.

Marshak, David S. *Open Electronic Mail*. **Open Information Systems** 8.6 (1993): 3–22.

Marshak, Ronni T. *BeyondMail for Windows*. **Office Computing Report** 15.9 (1992): 5–19.

Marshall, Chris & Nelson, Catherine & Gardiner, Margaret M. *Design Guidelines*. **Applying cognitive psychology to user-interface design**. Gardiner, Margaret M & Bruce, Christie, eds. John Wiley & Sons (1987): 221–278.

Martin, Rob. *Towards an Integrated Messaging Environment*. **Telecommunications** (Oct 1992): 54–59.

May, Brian. *X/Open's XDS & XOM Application Program Interfaces—An Implementors Experience*. **Proceedings of NetWorkShop92**, University of Queensland, Dec 1992.

Mayhew, Deborah J. *Principles and Guidelines in Software User Interface Design*. New Jersey: Prentice Hall, 1992.

McIntosh, Stephen & Piper, Tony, & Kaplan, Ilana & Cheung, Adrian. *Usability Testing a GUI Mail Application*. **Proceedings of OZCHI91—Australian Annual CHI Conference**, Hammond *et al*, eds. Sydney, Australia (1991): 103–107.

Microsoft Corporation. *The Microsoft Electronic Messaging Strategy*. Strategic White Paper, Microsoft Corporation Australia, 1992.

Mierswa, Peter O. *The Evolution of the MAILbus*. **Digital Technical Journal** 9 (June 1989): 37–43.

Miller, James R & Jeffries, Robin. *Usability Evaluation: Science of Trade-offs*. **IEEE Software** (Sept 1992): 97+

Miller, Richard & Moore, Christopher & White, James. *Closeup on APIs for X.400, a First Look*. **Data Communications** (June 21, 1990): 47-56.

Mitchell, C. *Multi-destination secure electronic mail*. **The Computer Journal** 32.1 (1989): 13–15.

Mitchell, Christopher & Walker, Michael & Rush, David. *CCITT/ISO Standards for Secure Message Handling*. **IEEE Journal on Selected Areas in Communications** 7.4 (May 1989): 517–524.

Mitchell, Chris J. *OSI and X.400 Security*. **Telecommunications** (May 1990): 49–54.

Mitchell, C & Rush, D & Walker, M. *A Secure Messaging Architecture Implementing the X.400–1988 Security Features*. **The Computer Journal** 33.4 (1990): 290–295.

Modley, Rudolf. *Handbook of Pictorial Symbols: 3250 Examples from International Sources*. New York: Dover Publications, 1976.

Molesworth, Roger. *The essentials of electronic directories*. **Proceedings of Electronic Messaging and Communications Systems 90**. London: Blenheim Online (Dec 1990): 95–102.

Molich, Rolf & Nielsen, Jakob. *Improving a Human-Computer Dialogue*. **Communications of the ACM** 33.3 (Mar 1990): 338–348.

Myers, Brad A. *The Importance of Percent-Done Progress Indicators for Computer-Human Interfaces*. **Proceedings of CHI'85**. (1985): 11–17.

Myers, Brad A. *Gaining General Acceptance for UIMS*. **Computer Graphics** 21.2 (Apr 1987): 130–134.

Myers, Brad A. *Creating User Interfaces by Demonstration*. San Diego: Academic Press, 1988.

Myers, Brad A. *User-Interface Tools: Introduction and Survey*. **IEEE Software** (Jan 1989): 15–23.

Myers, Brad A. *A New Model for Handling Input*. **ACM Transactions on Information Systems** 8.3 (July 1990): 289–320.

Myers, Brad A. *Creating User Interfaces Using Programming by Example, Visual Programming, and Constraints*. **ACM Transactions on Programming Languages and Systems** 12.2 (Apr 1990): 143–177.

Myers, Brad A. & Rosson, Mary Beth. *Survey on User Interface Programming*. **Proceedings of CHI'92,** Monterey, USA, 3–7 May. (1992): 195–202.

Naffah, Najah & Texier, Michel & Jureidini, Gabriel. *Intelligent User Interfaces for Advanced Workstations*. **Information Processing 89—Proceedings of IFIP 11th World Computer Congress**. Ritter, G X, ed. Elsevier Science Publishers. (1989): 1021–1024.

Navarro, Leandro. *Advanced User Interfaces for Distributed Group Communication*. **Human-Computer Interaction—INTERACT'90** Diaper, D *et al*, eds. Elsevier Science Publishers (1990): 1025–1027.

Neelamkavil, F & Teo G S. *X versus Eiffel toolkits for building graphical user interfaces*. **Information and Software Technology** 33.8 (Oct 1991): 559–565.

Neilson, I & Weir, G. *Dialogue Design and Delivery Systems*. Department of Computer Science Report No AMU–87–23. University of Strathclyde, UK, 1987.

Neufeld, Gerald & Brachman, Barry & Goldberg, Murray & Stickings, Duncan. *The EAN X.500 Directory Service*. **Internetworking: Research and Experience** 3 (1992): 55–81.

Neumann, Peter G. *Aegis, Vincennes, and the Iranian Airbus*. **ACM Software Engineering Notes** 14.5 (Jul 1989): 20–22.

Nielsen, Jakob. *Executive Summary: Coordinating User Interfaces for Consistency*. **Coordinating User Interfaces for Consistency**, Nielsen, Jakob ed. Academic Press (1989): 1–7.

Nielsen, Jakob. *Usability Engineering at a Discount*. **Designing and Using Human-Computer Interfaces and Knowledge Based Systems**. Salvendy, G & Smith, M J, eds. Elseivier Science Publishers (1989): 394–401.

Nielsen, J. *Usability Testing of International Interfaces*. **Designing User Interfaces for International Use**. Nielsen, J ed. Amsterdam: Elsevier Science Publications (1990): 39–44.

Nielsen, Jakob. *Paper versus Computer Implementations as Mockup Scenarios for Heuristic Evaluation*. **Human-Computer Interaction—INTERACT'90**. Diaper, D, *et al*, eds. Elsevier Science Publishers (1990): 315–320.

Nielsen, Jakob. *A meta-model for interacting with computers*. **Interacting with Computers** 2.2 (1990): 147–160.

Nielsen, Jakob. *The Usability Engineering Life Cycle*. **Computer** 25.3 (Mar 1992): 12–22.

Norman, Donald A. *Design rules Based on Analyses of Human Error.* **Communications of the ACM** 26.4 (April 1983): 254–258.

Norman, Donald A. *Cognitive Engineering.* **User Centered System Design.** Chapter 3. Norman, Donald A & Draper, Stephen W, eds. New Jersey: Lawrence Erlbaum Associates, 1986.

Norman, Donald A. *Commentary: Human Error And The Design of Computer systems.* **Communications of the ACM** 33.1 (Jan 1990): 4–7.

Norman, Donald A. *Cognitive Artifacts*. **Designing Interaction: Psychology and the Human-Computer Interface**. Carroll, John M, ed. New York: Cambridge University Press (1991):17–38.

Norman, Donald A. *Name confusion and it's implications*. **ACM Forum on Risks to the Public in Computers and Related Systems** 14.16 & 14.17, 8 Dec 1992.

Norman, Donald A *&* Draper, Stephen W. *User Centered System Design***.** New Jersey: Lawrence Erlbaum Associates, 1986.

Norman, Kent L. *The Psychology of Menu Selection: Designing Cognitive Control of the Human/Computer Interface.* New Jersey: Ablex Publishing, 1991.

Norman, Kent L & Weldon, Linda J & Shneiderman, Ben. *Cognitive layouts of windows and multiple screens for user interfaces.* **International Journal of Man-Machine Studies** 25 (1986): 229–248.

Ohmura, Haruki & Kamiyama, Yuichi & Kobayashi, Hiroshi. *Development of a multimedia MHS based on CCITT X.400 Recommendations*. **Computer Message Systems**. Uhlig, R, ed. Elsevier Science Publishers (1986): 305-319.

Olsen, Dan R Jr. *Larger Issues in User Interface Management.* **Computer Graphics** 21.2 (Apr 1987): 134–137.

Olsen, Dan R Jr. *User Interface Management Systems: Models and Algorithms*. Morgan Kaufmann Publishers, 1992.

Olsen, Dan R Jr & Dempsey, Elizabeth P & Rogge, Roy. *Input/Output Linkage in a User Interface Management System.* **Proceedings of SIGGRAPH'85** 19.3 (Jul 1985): 191–197.

Olsen, Dan R Jr & Foley, James D & Hudson, Scott E & Miller, James & Myers, Brad. *Research directions for user interface software tools*. **Behaviour & Information Technology** 12.2 (1993): 80–97.

Omnicom. *OSI Object Management API Specification* Version 2.0; *X.400 API Specification* Version 2.0; *X/Open Directory Services API Specification* Version 1.0. X.400 API Association & X/Open Company Ltd. Omnicom Inc USA, 1990.

OSF. *OSF/Motif User's Guide. OSF/Motif Style Guide. OSF/Motif Programmers Guide*. Open Software Foundation 1990.

OSF. *Directory Services for a Distributed Computing Environment*. White Paper OSF–O–WP9–0990–2, Open Software Foundation, Sept 1990.

Palme, Jacob. *Standards for asynchronous group communication*. **Computer Communication** 16.9 (Sep 1993): 532–538.

Pangalos, G J. *Standardization of the user interface*. **Computer Standards & Interfaces** 14 (1992): 223–229.

Pangalos, G J. *Consistency and standardization of user interfaces.* **Information and Software Technology** 34.6 (June 1992): 397–401.

Payne, Alison. X.*500 Directory Services Overview*. Technical Report No 182. Key Centre for Software Technology, The University of Queensland, Aug 1990.

Perlman, Gary. *An Overview of SAM: A Hypertext Interface to Smith & Mosier's Guidelines for Designing User Interface Software.* Technical Report TR-87-09, School of Information Technology, Wang Institute of Graduate Studies, Tyngsboro, MA 01879 USA. July 1987.

Perlman, Gary. *Evaluating how your user interfaces are used.* **IEEE Software**, Human Factors column (Jan 1989): 112–113.

Perlman, Gary. *A Vision of Universal Functionality for Tomorrow's User Interfaces*. **Proceedings of OZCHI92**, Rees, M J & Iannella, R, eds. Gold Coast, Australia, 26–27 Nov (1992): 1-14.

Perlman, Gary. *Practical User Interface Evaluation*. **OZCHI92 Conference Workshop**. Gold Coast, Australia, 25 Nov (1992).

Perlman, Gary & Moorhead, Tony. *NaviText™ SAM Software User Manual*. Northern Lights Software Corporation, Westford, MA 01886, USA. 1988.

Perry, Tekla S & Voelcker, John. *Of mice and menus: designing the user-friendly interface.* **IEEE Spectrum**. (Sept 1989): 46–51.

Persky, Jim. *Directory Services for the Macintosh*. **Connections** 6.1 (Jan 1992): 22–24.

Plaisant, Catherine & Shneiderman, Ben. *Scheduling home control devices: design issues and usability evaluation of four touchscreen interfaces*. **International Journal Man-Machine Studies** 36. (1992): 375–393.

Plattner, Bernhard & Steinparz, Franz X. *Electronic Message Processing Aspects of Filing and Retrieving Messages*. **Message Handling Systems**. Speth, R, ed. Elsevier Science Publishers (1988): 277–287.

Plattner, Bernhard & Lubich, Hannes. *Electronic Mail Systems and Protocols Overview and Case Study*. **Message Handling Systems and Distributed Systems**. Stefferud, E *et al*, eds. Elsevier Science Publishers (1989): 55–99.

Pliskin, Nava. *Interfacing with electronic mail can be a dream or a nightmare: a user's point of view*. **Interacting with Computers** 1.3 (1989): 259–272,

Pollock, Stephen. *A Rule-Based Message Filtering System*. **ACM Transactions on Office Information Systems** 6.3 (July 1988): 232–254.

Postel, Jonathan B. *Simple Mail Transfer Protocol.* **Request For Comments 821**, Aug 1982.

Postel, Jonathan B & Finn, Gregory G & Katz, Alan R and Reynolds, Joyce K. *An Experimental Multimedia Mail System*. **ACM Transactions on Office Information Systems** 6.1 (Jan 1988): 63–81.

Potosnak, Kathleen. *Getting the most out of design guidelines.* **IEEE Software**, Human Factors column (Jan 1988): 85–86.

Potosnak, Kathleen. *Setting objectives for measurably better software.* **IEEE Software**, Human Factors column (Mar 1988): 89–90.

Potosnak, Kathleen. *10 tips for getting useful information from users.* **IEEE Software**, Human Factors column (July 1988): 89–90.

Potosnak, Kathleen. *What's wrong with standard user interfaces?* **IEEE Software**, Human Factors column (Sept 1988): 91–92.

Potosnak, Kathleen. *Recipe for a usability test.* **IEEE Software**, Human Factors column (Nov 1988): 83–84.

Potosnak, Kathleen. *Management: The key to success.* **IEEE Software**, Human Factors column (Mar 1989): 86–88.

Potosnak, Kathleen. *Modular implementation benefits developers, users.* **IEEE Software**, Human Factors column (May 1989): 91,105.

Potosnak, Kathleen. *When a usability test is not the answer.* **IEEE Software**, Human Factors column (July 1989): 105–106.

Potosnak, Kathleen. *Mental models: Helping users understand software.* **IEEE Software**, Human Factors column (Sept 1989): 85–88.

Prasad, K V K K. *On the User Interfaces for Electronic Mail Systems*. **Telematics and Informatics** 7.2 (1990): 145–149.

Prime, Martin. *User Interface Management Systems—A Current Product Review*. **Computer Graphics Forum** 9 (1990): 53–76

Radicati, Sara. *Electronic Mail: An Introduction to the X.400 Message Handling Standards*. New York: MacGraw Hill, 1992.

Radicati, Sara. *The 1992 Extensions to X.500*. **ConneXions** 6.9 (Sept 1992): 2–9

Ravden, Susannah J & Johnson, Graham I. *Evaluating Usability of Human-Computer Interfaces: A Practical Method*. Ellis Horwood Limited, 1989.

Rees, Michael J. *Evolution and Evaluation of Graphical User Interfaces for Electronic Mail*. **Proceedings of OZCHI91**—Australian Annual CHI Conference, Hammond *et al*, eds. Sydney, Australia (1991): 53–58.

Rees, M J & Iannella, R. *COREmail: A fully functional electronic messaging system*. School of Information and Computing Sciences Working Paper 1990-3-025, Bond University, Australia. Feb 1990.

Rees, M J & Iannella, R. *QuickMail Usage Analysis*. School of Information and Computing Sciences Working Paper 1990-3-039, Bond University, Australia. July 1990.

Rees, Michael J & Iannella, Renato. *Design Strategies for Graphical User Interfaces to X.400 Electronic Mail*. **Proceedings of Australian Computing Conference**, Gold Coast, Australia 5-8 Sept. Australian Computer Society (1990): 97–106.

Reinhardt, Andy. *Smarter E-Mail is Coming*. **BYTE** (Mar 1993): 90–108.

Rhyne, Jim. *Dialogue Management for Gestural Interfaces*. **Computer Graphics** 21.2 (Apr 1987): 137–142.

Rhyne, Jim & Ehrich, Roger & Bennett, John & Hewett, Tom & Sibert, John & Bleser, Terry. *Tools and Methodology for User Interface Development.* **Computer Graphics** 21.2 (Apr 1987): 78–87.

Robbins, Colin J. *Managing the International X.500 Directory Pilot*. **Proceedings of EurOpen Autumn '91**, Budapest (Sept 1991): 187–197.

Robbins, Colin J & Kille, Stephen E. *The ISO Development Environment: User's Manual. Volume 5: QUIPU*. University College London & X-Tel Services Ltd. 1991.

Rodden, T & Sawyer, P & Sommerville, I. *Vista: a user interface for a distributed object-oriented software engineering environment.* **Software Engineering Journal**. (Jan 1992): 25–34.

Rogers, Yvonne. *Icon Design for the User Interface.* **International Reviews of Ergonomics** 2. 1989: 129–154.

Rose, Marshall T. *The ISO Development Environment: User's Manual*. Performance Systems International Inc, 1990.

Rose, M. *Directory Assistance Service*. **Request for Comments 1202** Feb 1991.

Rose, Marshall T. *The Little Black Book: Mail Bonding with OSI Directory Services*. Prentice Hall, 1992.

Rosenberg, Daniel. *A Cost Benefit Analysis for Corporate User Interface Standards: What price to pay for a consistent "look and feel"?* **Coordinating User Interfaces for Consistency**, Nielsen, Jakob ed. Academic Press (1989): 21–34.

Rosenthal, David. *A Simple X11 Client Program—or—How hard can it really be to write "Hello, World"?* **Proceedings of USENIX Winter Conference**, Dallas, Texas (Feb 1988): 229–242.

Rovira, Charles A. *Rovira Diagrams*. **Computer Language** (Jan 1990): 59–67.

Rowe, Lawrence A & Konstan, Joe & Smith, Brian & Seitz, Steve & Liu, Chung. *The Picasso Application Framework.* Technical Report, University of California. 1990.

Rubin, Tony. *Human factors in computer based message systems*. **Computer Bulletin** (Sept 1986): 16–18.

Rubin, T. *User Interface Design for Computer Systems.* England: Ellis Horwood Limited, 1988

Salamone, Salvatore. *Merging the Macintosh into Enterprise Nets*. **Data Communications** (Nov 1992): 49–50.

Sastry, Lakshmi. *User Interface Management Systems for Engineering Applications.* **Computer Graphics Forum** 11.2 (1992): 113–129.

Schaffrina, Jorg. *Interrelations of Industrial Design, Ergonomics, and the User*. **IEEE Journal on Selected Areas in Communications** 9.4 (May 1991): 501–505.

Scheifler, R W & Gettys, J. *The X Window System*. **ACM Transactions on Graphics** 5.2 (Apr 1987): 79–109.

Schicker, Pietro. *Message Handling Systems X400*. **Message Handling Systems and Distributed Applications**. Stefferud, E *et al*, eds. Elsevier Science Publishers (1989): 3–41.

Scott, Karyl. *X.400 Pushes the Envelope for Electronic Messaging*. **Data Communications** (June 1990): 93–102.

Shackel, B. *Interface Design for Usability*. **User Interfaces**. Bernold, T, ed. Elsevier Science Publishers (1988): 59–70.

Shan, Yen-Ping. *An Object-Oriented UIMS for Rapid Prototyping.* **Human-Computer Interaction—INTERACT'90** Diaper, D *et al*, eds. Elsevier Science Publishers (1990): 633–638.

Shneiderman, Ben, *Seven plus or minus two central issues in human-computer interaction.* **Proceedings of CHI'86,** Boston, USA. (1986): 343–349.

Shneiderman, Ben. *Intelligent Interfaces: From fantasy to fact.* **Information Processing 89—Proceedings of IFIP 11th World Computer Congress**. Ritter, G X, ed. Elsevier Science Publishers. (1989): 915.

Shneiderman, Ben. *Future Directions for Human-Computer Interaction.* **International Journal of Human-Computer Interaction** 2.1 (1990): 73-90.

Sibert, John L & Hurley, William D & Bleser, Teresa W. *An Object-Oriented User Interface Management System.* **Proceedings of SIGGRAPH'86** 20.4 (Aug 1986): 259–268.

Siebel, Mike & Crispin, Mark & Lundblade, Laurence. *Pine Technical Notes*. University of Washington, July 1992.

Sinderen, Marten van & Dorregeest, Evert. *A Critical Analysis of the X.400 Model of Message Handling Systems*. **Computer Standards & Interfaces** 7 (1988): 363–375.

Singh, Gurminder & Green, Mark. *Automating the Lexical and Syntactic Design of Graphical User Interfaces: The UofA* UIMS*. **ACM Transactions on Graphics** 10.3 (Jul 1991): 213–254.

Siochi, Antonio C & Ehrich, Richard W. *Computer Analysis of User Interfaces Based on Repetition in Transcripts of User Sessions*. **ACM Transactions on Information Systems** 9.4 (Oct 1991): 309–335.

Skogseth, Gunn & Verne, Guri. *Requirements in a Multimedia Document System*. **Computer Message Systems–85**. Uhlig, R, ed. Elsevier Science Publishers (1986): 221–233.

Smith, Sidney L & Mosier, Jane N. *Guidelines for Designing User Interface Software.* The MITRE Corporation, Bedford, MA 01730-0208, USA. Report: ESD-TR-86-278 MTR-10090. 1986.

Smith, Timothy William. *Assessing the Usability of User Interfaces: Guidance and Online Help Features*. Doctoral Thesis. Department of Business Administration, The University of Arizona, 1988.

Sonnenwald, Diane H. *An Architecture for Human-Network Interfaces.* **IEEE Network Magazine.** (Nov 1990): 61–69.

Standards Australia. *Naming and Addressing in the Australian OSI Environment*. Publication SAA MP59–1991, Standards Australia, 1991.

Steedman, Douglas. *X.500 The directory standard and its application*. Twickenham, UK: Technology Appraisals, 1993.

Stern, Richard H. A*ppropriate and inappropriate legal protection of user interfaces and screen displays.* **IEEE Micro** (June 1989): 84–88.

Stewart, Tom. *Standards as a Means of Influencing Interface Design.* **Human Aspects in Computing: Design and Use of Interactive Systems and Work with Terminals.** Bullinger, H J, ed. Elsevier Science Publishers (1991): 538–542.

Sykas, E D & Lyberopoulos, G L. *Overview of the CCITT X.500 recommendations series*. **Computer Communications** 14.9 (Nov 1991): 545–556.

Szczur, Martha A & Sheppard, Sylvia B. *TAE Plus: Transportable Applications Environment Plus: A User Interface Development Environment.* **ACM Transactions on Information Systems** 11.1 (Jan 1993): 76–101.

Tandem Computers. *Electronic Commerce: A Strategic Advantage*. Tandem Computers Inc, 1992.

Tarouco, Liane Margarida Rockenbach. *User Friendly Interface for Messaging Systems*. **Computer Based Message Services**. Smith, H T, ed. Elsevier Science Publishers (1984): 167–174.

Taylor, Trevor. *A GUI Review.* **ON$DECK Magazine** (Jul/Aug 1991): 38–46.

Thimbleby, Harold. *Can anyone work the video?* **New Scientist**, (23 Feb 1991): 48–51.

Thomas, John C & Kellogg, Wendy A. *Minimizing Ecological Gaps in Interface Design.* **IEEE Software**. (Jan 1989): 78–86.

Thomas, Robert H & Forsdick, Harry C & Crowley, Terrence R & Schaaf, Richard W & Tomlinson, Raymond S & Travers, Virginia M & Robertson, George G. *Diamond: A Multimedia Message System Built on a Distributed Architecture*. **Computer** (Dec 1985): 65–77.

Thompson, Ken. *Human Interfaces Workshop Report*. **Message Handling Systems**. Speth, R, ed. Elsevier Science Publishers (1988): 327.

Tognazzini, Bruce. *TOG on Interface*. Addison-Wesley, 1992.

Toleman, Mark A & Welsh, Jim. *Retrospective Application Of User Interface Guidelines: A Case Study Of A Language-Based Editor.* **Proceedings of OZCHI91**, Sydney, 28–29 Nov, (1991): 33–38.

Treadway, Richard, *Separating Form from Function in Future Interface Management Technologies*. **NCGA'89 Conference Proceedings** 2. National Computer Graphics Association USA, (1989): 239–247.

Tse, E K & Najjar, M & Kawaguchi, K & Yamamoto, Y. *Interconnection of Two Message Handling Systems Based on CCITT X.400 Series Recommendations*. **Computer Standards & Interfaces** 7 (1988): 11-16.

Tyldesley, D A. *Employing Usability Engineering in the Development of Office Products*. **The Computer Journal** 31.5 (1988): 431–436.

Uhlir, Steven. *Enabling the user interface*. **IBM Systems Journal** 27.3 (1988): 306–314.

Ullmann, Robert. *Mapping between Internet and X.400 addresses*. Draft Request for Comments, July 1989.

Updegrove, Daniel A & Muffo, John A & Dunn, John A Jr. *Electronic Mail And Networks: New Tools For University Administration*. **EDUCOM Review** (Spring 1990): 21–28.

US Department of Defense. *Military Standard: Human Engineering Design Criteria for Military Systems, Equipment, and Facilities*. MIL-STD-1472D, US Department of Defense, Washington, 1989.

Vainio-Larsson, Arja & Orring, Rebecca. *Evaluating the Usability of User Interfaces: Research in Practice*. **Human-Computer Interaction—INTERACT'90**. Diaper, D, *et al*, eds. Elsevier Science Publishers (1990): 323–328.

Vervest, Peter. *Electronic Mail and Message Handling*. London: Francis Pinter Publishers, 1985.

Vervest, Peter H M. *Innovation In Electronic Mail: Towards Open Information Networks—Perspectives on Innovation Policy*. North Holland Publishers, 1987.

Waddington, Ray & Johnson, Peter. *Designing and Evaluating Interfaces Using Task Models*. **Information Processing 89**. Ritter, G X, ed. Elsevier Science Publishers (1989): 247–252.

Wahlster, Wolfgang. *Intelligent Interfaces as Cooperative Agents.* **Information Processing 89—Proceedings of IFIP 11th World Computer Congress**. Ritter, G X, ed. Elsevier Science Publishers. (1989): 917.

Walravens, Jean. *X.400—The Message Handling System Standard*. **1992 Single Market Communication Review** 3.2 (July 1991): 96–101.

Wasserman, Anthony I & Shewmake, David T. *The role of Prototypes in the User Software Engineering (USE) Methodology*. **Human-Computer Interaction**: **Selected Readings**, Preece, Jenny & Keller, Laurie & Stolk, Hans, eds. Prentice Hall (1990): 385–401.

Waugh, Andrew. *Obtaining, Establishing, & Operating an X.500 Directory*. **NetWorkShop92 Tutorial**, University of Queensland, Dec 1992.

Weider, Chris & Wright, Russ. *A Survey of Advanced Usages of X.500*. **Integrated Directory Services Working Group Internet Draft**, June 1993.

Weir, George R S & Qing, Ni Xiao. *Second Language User Support*. Department of Computer Science Technical Report. University of Strathclyde, UK 1993.

Whitefield A & Sutcliffe A. *Case study in human factors evaluation.* **Information and Software Technology** 34.7 (July 1992): 443–451.

Whiteside, John & Bennett, John & Holtzblatt. Karen. *Usability Engineering: Our Experience and Evolution.* **Handbook of Human-Computer Interaction**, Helander, M, ed. Elsevier Science Publishers (1988): 791–817.

Whitten, David. *X.400: Breaking Vendor Boundaries For Enterprise-Wide E-Mail*. **Telecommunications** (July 1989): 47–52.

Whitten, Emma. *E-Mail: The Corporate Choice?* **Telecommunications** (Mar 1993): 30–32.

Wickham, G J. *Mapping Electronic Mail Addresses using X.500*. **Proceedings of NetWorkShop '91**, University of Tasmania, 1991.

Wickham, Greg. *The Characteristics of the X.500 Directory Interface for Locating and Extracting the Entries of an Individual*. CSIRO Division of Information Technology, Australia. March 1992.

Wiecha, Charles. *ITS and User Interface Consistency: A Response to Grudin.* **ACM Transactions on Information Systems** 10.1 (Jan 1992): 112-114.

Wildman, Luke & Hayes, Ian. *A Formal Specification of X.500: Abstract Directory Service Definition*. Key Centre for Software Technology Technical Report 183, University of Queensland, Sept 1990.

Williamson, Christopher & Schneiderman, Ben. *The Dynamic HomeFinder: Evaluating Dynamic Queries in a Real-Estate Information Exploration System.* **SIGIR Forum** (June 1992): 338–346.

Wilson, Michael & Conway, Anthony. *Enhanced Interaction Styles for User Interfaces*. **IEEE Computer Graphics & Applications** (Mar 1991): 79–90.

Wood, Catherine A & Gray, Philip D. *User Interface-Application Communication in the Chimera User Interface Management System*. **Software-Practice and Experience** 22.1 (Jan 1992): 63–84.

Wright, Peter C & Monk, Andrew F. *A cost-effective evaluation method for use by designers*. **International Journal of Man-Machine Studies** 35 (1991): 891–912.

Yankelovich, Nicole & Haan, Bernard J & Meyrowitz, Norman K & Drucker, Steven M. *Intermedia: The Concept and the Construction of a Seamless Information Environment*. **Computer** (Jan 1988): 81–96.

Yeong, Wengyik & Howes, Tim & Hardcastle-Kille, Steve. *Lightweight Directory Access Protocol*. Network Working Group Internet Draft, Dec 1992.

Zahm, A. *An X.500 Schema for Hotels and Restaurants*. **OSI-Directory Services Internet Draft 35**, Oct 1992.

Zainlinger, Ralph. *Building Interfaces for CASE Environments: An Object Oriented Interaction Model and its Application*. **Human factors in information systems analysis and design**. Finkelstein, A & Tauber, M J & Traunmuller R, eds. Elsevier Science Publishers, 1990: 65–80.

Zhou, Tom Z.Y. & Kubitz, William J. *An Object-Oriented View of the User Interface.* **Eurographics 92** 11.3 Kilgour, A & Kjelldahl, L, eds. (1992): 81–92.